aws

Entwicklerhandbuch

Amazon DocumentDB



Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon DocumentDB: Entwicklerhandbuch

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Die Handelsmarken und die Handelsaufmachung von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, durch die Kunden irregeführt werden könnten oder Amazon in schlechtem Licht dargestellt oder diskreditiert werden könnte. Alle anderen Handelsmarken, die nicht Eigentum von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise zu Amazon gehören oder nicht, mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Was ist Amazon DocumentDB ... 1
-Übersicht ... 1
Cluster ... 3
Instances ... 4
Regionen und AZs ... 8
Regionen ... 8
Availability Zones ... 8
Preisgestaltung ... 11
Kostenlose Testversion ... 12
Überwachen ... 12
Schnittstellen ... 12
AWS-Managementkonsole ... 12
AWS CLI ... 13
MongoDB-Treiber ... 13
Als nächstes ... 13
Funktionsweise ... 14
Amazon DocumentDB-Versionen ... 14
Amazon DocumentDB-Endpunkte ... 17
TLS-Unterstützung ... 20
Amazon DocumentDB-Speicher ... 21
Amazon DocumentDB-Replikation ... 22
Zuverlässigkeit von Amazon DocumentDB ... 22
Präferenzoptionen lesen ... 23
TTL löscht ... 28
Abrechnungsfähige Ressourcen ... 29
Was ist eine Dokumentendatenbank? ... 32
Anwendungsfälle ... 33
Dokumente verstehen ... 34
Arbeiten mit Dokumenten ... 40
Leitfaden für die ersten Schritte ... 53
Voraussetzungen ... 54
Schritt 1: Erstellen eines Clusters ... 55
Schritt 2: Stellen Sie eine Verbindung zu Ihrem Cluster her ... 58
Schritt 3: Daten einfügen und abfragen ... 59

Schritt 4: Erkunden ... 61
Schneller Einstieg in die Verwendung CloudFormation ... 62
Voraussetzungen ... 62
Eine Amazon DocumentDB starten CloudFormation Stack ... 63
Zugreifen auf den Amazon DocumentDB-Cluster ... 69
Kündigungsschutz und Löschschutz ... 69
Migration und Aktualisierung ... 71
Schnellstart-Anleitung ... 71
Bereiten Sie die DMS-Quelle vor ... 72
DMS einrichten ... 72
Aktivieren Sie die DocumentDB-Komprimierung ... 72
Erstellen Sie eine Replikationsaufgabe ... 72
Überwachen des Fortschritts ... 73
Zusätzliche Informationen ... 74
Runbook zur Migration ... 74
Kompatibilität ... 75
Workload-Erkennung ... 78
Neuen Zielcluster erstellen ... 81
Index-Migration ... 82
Benutzermigration ... 83
Datenmigrationen ... 85
Überwachen ... 96
Validierung ... 99
Migration von Couchbase Server ... 104
Einführung ... 105
Vergleich mit Amazon DocumentDB ... 105
Erkennung ... 108
Planung ... 113
Migration ... 116
Validierung ... 125
Termine für die Unterstützung der Engine-Version ... 128
Veröffentlichungskalender für Amazon DocumentDB DocumentDB-Hauptversionen ... 129
Veröffentlichungskalender für Amazon DocumentDB-Nebenversionen ... 129
Aktualisierung einer Hauptversion der Engine ... 130
Unterstützte Upgrade-Pfade ... 131
Voraussetzungen ... 132

Bewährte Methoden 134
Was ändert sich nach dem Upgrade auf Amazon DocumentDB 8.0 137
Post-upgrade Überlegungen zu Clustern, die von 3.6 oder 4.0 aktualisiert wurden 138
Durchführung des Upgrades 139
Fehlerbehebung 141
Aktualisierung einer kleineren Engine-Version 142
Vor dem Durchführen eines Nebenversions-Upgrades 142
Durchführen eines Nebenversions-Upgrades 143
Das Upgrade der Nebenversion wird überprüft 144
Ein Upgrade durchführen mit AWS DMS 144
Schritt 1: Change Streams aktivieren 145
Schritt 2: Ändern Sie die Aufbewahrungsdauer für Change-Streams 145
Schritt 3: Migrieren Sie Ihre Indizes 145
Schritt 4: Erstellen Sie ein AWS DMS replication instance 147
Schritt 5: Erstellen Sie eine AWS DMS Quell-Endpunkt 149
Schritt 6: Erstellen Sie eine AWS DMS Zielendpunkt 151
Schritt 7: Erstellen Sie eine Migrationsaufgabe und führen Sie sie aus 153
Schritt 8: Ändern des Anwendungsendpunkts auf den Amazon DocumentDB DocumentDB-Zielcluster 155
Long-Term Support- (LTS, Langzeit-Support) Versionen nutzen 155
Aktuelle LTS-Version 155
Wer sollte LTS verwenden 156
Wie erstelle ich einen neuen LTS-Cluster 156
Wie führe ich ein Upgrade von Engine-Version 3.6.0 oder 4.0.0 auf einen 5.0.0 LTS-Cluster durch 157
Wie überprüfe ich, ob mein vorhandener Engine-Cluster mit Version 5.0.0 auf LTS läuft 157
Erweiterter Support 158
Überblick über den erweiterten Support 159
Gebühren für erweiterten Support 160
Verantwortlichkeiten 160
Funktionen und Konfigurationen 161
Status der Engine 161
Cluster-Typen 161
Features 162
Instance-Typen 163
AWS Integrationen 164

Auswählen von Regionen und Availability Zones ... 165
Verfügbarkeit in Regionen ... 166
OpenSearch Integration ... 169
Amazon OpenSearch Service als Ziel ... 169
Einschränkungen ... 173
Entwickeln ... 174
Mit DocumentDB Connect ... 174
Programmgesteuertes Herstellen einer Verbindung ... 175
Verbindung von außerhalb einer Amazon VPC herstellen ... 205
Als Replikatgruppe wird eine Verbindung hergestellt ... 207
Connect mit Studio 3T her ... 212
Connect mit DataGrip ... 222
Stellen Sie mithilfe von Amazon EC2 eine Verbindung her ... 227
Stellen Sie mithilfe des JDBC-Treibers eine Verbindung her ... 267
Stellen Sie mithilfe des ODBC-Treibers eine Verbindung her ... 284
Programm mit Amazon DocumentDB ... 298
DocumentDB Java-Programmierhandbuch ... 298
Verwenden der JSON-Schemavalidierung ... 332
Abfrage ... 343
Dokumente abfragen ... 343
Plan abfragen ... 346
Erklären Sie die Ergebnisse ... 348
Abfrageplaner v2 ... 354
Abfrageplaner v3 ... 369
Koordinatenbasierte Daten ... 374
Teilweiser Index ... 380
Textsuche ... 388
Indizes ... 396
Verwaltung von Indizes ... 396
Indextypen ... 408
Eigenschaften indizieren ... 417
Verwendung von Change Streams ... 436
Unterstützte Vorgänge ... 437
Erweiterte Ereignisse ... 438
Fakturierung ... 439
Einschränkungen ... 439

Change Streams aktivieren ... 440
Verwendung von Change-Streams mit Python ... 442
Vollständige Dokumentensuche ... 445
Verwenden Sie $changeStreamSplitLargeEvent , um große Ereignisse in Fragmente aufzuteilen ... 446
Wiederaufnahme eines Change-Streams ... 446
Wiederaufnahme mit `startAtOperationTime` ... 448
Wiederaufnahme mit `postBatchResumeToken` ... 450
Transaktionen ... 451
Änderung der Aufbewahrungsdauer des Protokolls ... 451
Change Streams auf sekundären Instanzen ... 455
Transaktionen ... 456
Voraussetzungen ... 457
Bewährte Methoden ... 457
Einschränkungen ... 458
Überwachung und Diagnose ... 459
Isolationsstufe der Transaktion ... 460
Anwendungsfälle ... 460
Unterstützte Befehle ... 498
Nicht unterstützte Funktionen ... 498
Sitzungen ... 499
Fehler bei der Transaktion ... 501
Wiederholbare Schreibvorgänge ... 502
Voraussetzungen ... 502
Unterstützte Vorgänge ... 502
Einschränkungen ... 503
Aktivieren ... 504
Funktionsweise ... 504
Fehlerbehandlung ... 505
Migration ... 505
Sortierung verwenden ... 506
Einschränkungen ... 506
Verwenden von -Ansichten ... 507
Bewährte Methoden ... 507
Kompatibilität mit Aggregator-Operatoren ... 508
Nutzung von Indizes und Ansichten ... 508

Generative KI ... 509
SageMaker Leinwand ... 509
Vektor-Suche ... 512
Verwenden AWS Lambda mit Change-Streams ... 525
Einschränkungen ... 526
MongoDB-Kompatibilität ... 527
MongoDB 8.0-Kompatibilität ... 527
Was ist neu in Amazon DocumentDB 8.0 ... 527
Beginnen Sie mit Amazon DocumentDB ... 528
Führen Sie ein Upgrade durch oder migrieren Sie zu Amazon DocumentDB 5.0 oder 8.0 ... 529
Funktionelle Unterschiede ... 529
MongoDB 5.0-Kompatibilität ... 530
Was ist neu in Amazon DocumentDB 5.0 ... 530
Was ist neu in Amazon DocumentDB 5.0.1 ... 531
Beginnen Sie mit Amazon DocumentDB 5.0 ... 528
Führen Sie ein Upgrade durch oder migrieren Sie zu Amazon DocumentDB 5.0 ... 529
Funktionelle Unterschiede ... 529
MongoDB 4.0-Kompatibilität ... 534
Funktionen von Amazon DocumentDB 4.0 ... 534
Beginnen Sie mit Amazon DocumentDB 4.0 ... 535
Führen Sie ein Upgrade durch oder migrieren Sie zu Amazon DocumentDB 4.0 ... 536
Funktionelle Unterschiede ... 536
Funktionale Unterschiede zu MongoDB ... 537
Funktionale Vorteile von Amazon DocumentDB ... 538
Die funktionalen Unterschiede wurden aktualisiert ... 539
Funktionale Unterschiede zu MongoDB ... 543
Unterstützte MongoDB-APIs, -Operationen und -Datentypen ... 552
Datenbank-Befehle ... 553
Abfrage- und Projektionsoperatoren ... 563
Operatoren aktualisieren ... 566
Geodaten ... 569
Cursor-Methoden ... 570
Betreiber von Aggregations-Pipelines ... 572
Datentypen ... 590
Indizes und Indexeigenschaften ... 592
Betreiber von Aggregations-Pipelines ... 593

Geodaten ... 1233
Abfrage- und Projektionsoperatoren ... 1258
Operatoren aktualisieren ... 1328
Verwaltung von Clustern und Instances ... 1383
Überblick über die betrieblichen Aufgaben ... 1383
Hinzufügen einer Replik zu einem Amazon DocumentDB-Cluster ... 1384
Beschreibung von Clustern und Instanzen ... 1385
Einen Cluster-Snapshot erstellen ... 1387
Wiederherstellung aus einem Snapshot ... 1388
Eine Instanz aus einem Cluster entfernen ... 1389
Löschen eines Clusters ... 1390
Globale Cluster ... 1390
Was ist ein globaler Cluster? ... 1390
Wie sind globale Cluster nützlich? ... 1391
Was sind die aktuellen Einschränkungen globaler Cluster? ... 1391
Schnellstart-Anleitung ... 1392
Verwaltung globaler Cluster ... 1405
Globale Cluster verbinden ... 1412
Überwachung globaler Cluster ... 1413
Notfallwiederherstellung ... 1414
Verwalten von Clustern ... 1440
Erläuterungen zu Clustern ... 1441
Cluster-Einstellungen ... 1444
Cluster-Speicherkonfigurationen ... 1446
Den Status eines Clusters ermitteln ... 1450
Cluster-Lebenszyklus ... 1451
Skalieren von Clustern ... 1496
Klonen eines Volumes für einen Cluster ... 1500
Grundlegendes zur Cluster-Fehlertoleranz ... 1514
Verwalten von Instances ... 1515
Den Status einer Instance ermitteln ... 1515
Instance-Lebenszyklus ... 1515
Instanzklassen verwalten ... 1540
NVMe-backed Instanzen ... 1553
Verwaltung von Subnetzgruppen ... 1556
Erstellen einer Subnetzgruppe ... 1557

Beschreibung einer Subnetzgruppe ... 1563
Ändern einer Subnetzgruppe ... 1565
Löschen einer Subnetzgruppe ... 1569
Hohe Verfügbarkeit und Replikation ... 1570
Skalierung von Lesevorgängen ... 1570
Hohe Verfügbarkeit ... 1571
Hinzufügen von -Replicas ... 1572
Failover ... 1573
Replikationsverzögerung ... 1578
Verwaltung von Ereignissen ... 1579
Veranstaltungskategorien anzeigen ... 1580
Amazon DocumentDB DocumentDB-Ereignisse anzeigen ... 1582
Verwaltung von Cluster-Parametergruppen ... 1585
Beschreibung der Cluster-Parametergruppen ... 1586
Cluster-Parametergruppen erstellen ... 1594
Cluster-Parametergruppen ändern ... 1597
Ändern von Clustern zur Verwendung benutzerdefinierter Cluster-Parametergruppen ... 1602
Cluster-Parametergruppen kopieren ... 1604
Cluster-Parametergruppen zurücksetzen ... 1606
Löschen von Cluster-Parametergruppen ... 1609
Referenz zu Cluster-Parametern ... 1612
Endpunkte verstehen ... 1636
Die Endpunkte eines Clusters finden ... 1637
Den Endpunkt einer Instanz finden ... 1639
Verbindung zu Endpunkten herstellen ... 1643
Grundlegendes zu Amazon DocumentDB DocumentDB-ARNs ... 1644
Konstruieren eines ARN ... 1645
Suchen eines ARNs ... 1649
Taggen von Ressourcen ... 1651
Übersicht über -Ressourcen-Tags ... 1651
Tag-Einschränkungen ... 1653
Hinzufügen oder Aktualisieren von Tags ... 1653
Auflisten von Tags ... 1655
Entfernen von Tags ... 1656
Wartung von Amazon DocumentDB ... 1658
Wartungsmaßnahmen für Amazon DocumentDB ... 1659

Versionsnummerierung ... 1660
Wartungsfenster verwalten ... 1661
Benachrichtigungen über Engine-Patches ... 1664
Ausstehende Wartungsarbeiten anzeigen ... 1665
Engine-Aktualisierungen ... 1669
Nebenversionen ... 1672
Betriebssystemupdates ... 1673
User-initiated aktualisiert ... 1677
Globale Cluster ... 1678
Grundlegendes zu serviceverknüpften Rollen ... 1679
Service-linked Rollenberechtigungen ... 1680
Erstellen einer serviceverknüpften Rolle ... 1682
Ändern einer serviceverknüpften Rolle ... 1682
Löschen einer serviceverknüpften Rolle ... 1682
Unterstützte Regionen für serviceverknüpfte Amazon DocumentDB-Rollen ... 1683
Leistungsoptimierung ... 1684
Allgemeine Leistungsprobleme ... 1684
Hohe CPU-Auslastung ... 1684
Verbindungsprobleme ... 1687
Die Abfrage läuft langsam ... 1694
Anti-patterns ... 1699
Zusammengesetzter Index mit Anti-Pattern und mehr als 3 Attributen ... 1699
Abfragen mit langer Laufzeit ... 1701
Wenig genutzte und redundante Indizes ... 1704
Multi-key Indizes mit großen Arrays ... 1707
Tipps zur Leistungsverbesserung ... 1708
1. Verwenden Sie $match als erste Phase in Aggregationspipelines ... 1708
2. Verwenden Sie $project in der Aggregationspipeline, um die Größe der Pipeline-Daten zu minimieren ... 1708
3. Aktivieren Sie die Dokumentenkomprimierung, um die Speicherkosten und Kosten zu I/O senken und die Abfrageleistung zu verbessern ... 1709
4. Nutzen Sie Indizes für eine optimale Abfrageleistung ... 1710
5. Optimieren Sie Datenmodelle auf der Grundlage von Abfragemustern ... 1711
Plananalyse abfragen ... 1713
Allgemeiner Zeitplan ... 1715
Phasen der Ausführung ... 1715

Grundlegendes zu DocsExamined in Explain Plan Execution Stats ... 1717
Index-only scannt ... 1725
Wenn Amazon DocumentDB einen reinen Index-Scan verwendet ... 1726
Anforderungen für eine abgedeckte Abfrage ... 1728
Einschränkungen ... 1729
Bestätigung eines reinen Index-Scans ... 1729
Bewährte Methoden ... 1731
Cache-Filter planen ... 1731
Unterstützte Befehle ... 1732
Unterstützte Abfrage-Shapes ... 1734
Filter setzen, auflisten und löschen ... 1734
Stellen Sie sicher, dass ein Filter angewendet wird ... 1736
Beispiele ... 1737
Verhalten bei Hinweisen und Indexänderungen ... 1740
Bewährte Methoden ... 1740
Grundlegende Anleitungen für den Betrieb ... 1741
Dimensionierung der Instanz ... 1742
Arbeiten mit Indizes ... 1744
Bewährte Methoden für die Gewährleistung der Sicherheit ... 1746
Kostenoptimierung ... 1746
Verwendung von Metriken zur Identifizierung von Problemen mit der Leistung ... 1747
TTL- und Zeitreihen-Workloads ... 1751
Migrationen ... 1752
Arbeiten mit Cluster-Parametergruppen ... 1752
Abfragen der Aggregationspipeline ... 1753
`BatchInsert und BatchUpdate` ... 1753
Verwalten von Indexen ... 1753
Erstellung eines Amazon DocumentDB-Indexes ... 1753
Pflege der Amazon DocumentDB-Indizes ... 1760
Verwaltung der Dokumentenkomprimierung ... 1762
Verwaltung der Dokumentenkomprimierung ... 1763
Überwachung der Dokumentenkomprimierung ... 1765
Verwaltung der wörterbuchbasierten Komprimierung ... 1766
Leistungsaspekte ... 1767
Aktivierung der wörterbuchbasierten Komprimierung ... 1767
Erste Schritte ... 1768

Überwachen 1768
Sicherheit 1770
Passwortverwaltung mit AWS Secrets Manager 1771
Einschränkungen bei der Secrets Manager-Integration mit Amazon DocumentDB 1772
Überblick über die Verwaltung von Primärbenutzerkennwörtern mit AWS Secrets Manager 1772
Durchsetzung der Verwaltung des primären Benutzerpassworts durch Amazon DocumentDB in AWS Secrets Manager 1773
Verwaltung des primären Benutzerpassworts für einen Cluster mit Secrets Manager 1774
Datenschutz 1774
Client-side Verschlüsselung auf Feldebene 1775
Verschlüsseln von Daten im Ruhezustand 1782
Verschlüsseln von Daten während der Übertragung 1791
Schlüsselverwaltung 1803
Identitäts- und Zugriffsverwaltung 1803
Zielgruppe 1804
Authentifizierung mit Identitäten 1804
Verwalten des Zugriffs mit Richtlinien 1806
Wie Amazon DocumentDB mit IAM funktioniert 1807
Identity-based Beispiele für Richtlinien 1814
Fehlerbehebung 1818
Verwaltung der Zugriffsberechtigungen für Ihre Amazon DocumentDB-Ressourcen 1820
Verwenden von identitätsbasierten Richtlinien (IAM-Richtlinien) 1826
AWS verwaltete Richtlinien für Amazon DocumentDB 1830
Referenz zu den Amazon DocumentDB-API-Berechtigungen 1850
Cross-service verwirrt, stellvertretender Vertreter der Prävention 1859
Authentifizierung mithilfe der IAM-Identität 1860
Erste Schritte mit IAM-Benutzern und -Rollen 1861
Compute-Typen konfigurieren 1864
Überwachung von Authentifizierungsanfragen 1865
Verwenden der IAM-Authentifizierung 1866
Treiber, die IAM unterstützen 1866
Häufig gestellte Fragen zur IAM-Identitätsauthentifizierung 1867
Authentifizierung mit SCRAM-SHA-256 1867
Erstellen von -Benutzern 1868
Aktualisieren von Benutzern 1869

Authentifizieren ... 1870
Verwaltung von Amazon DocumentDB-Benutzern ... 1870
Primär- und `ServiceAdmin-Benutzer` ... 1870
Zusätzliche Benutzer erstellen ... 1871
Automatisches Rotieren von Passwörtern ... 1873
Role-Based Zugriffskontrolle ... 1874
RBAC-Konzepte ... 1875
Erste Schritte mit den integrierten RBAC-Rollen ... 1877
Erste Schritte mit benutzerdefinierten RBAC-Rollen ... 1881
Stellen Sie als Benutzer eine Verbindung zu Amazon DocumentDB her ... 1885
Allgemeine Befehle ... 1887
Funktionale Unterschiede ... 1892
Einschränkungen ... 1892
Datenbankzugriff mit Role-Based Access Control ... 1893
Aktualisierung der Zertifikate ... 1901
Aktualisierung Ihrer Anwendung und Ihres Amazon DocumentDB-Clusters ... 1902
Automatische Rotation von Serverzertifikaten ... 1906
Fehlerbehebung ... 1908
Häufig gestellte Fragen ... 1908
Compliance-Validierung ... 1915
Ausfallsicherheit ... 1916
Sicherheit der Infrastruktur ... 1917
VPC-Endpunkte (AWS PrivateLink) ... 1918
Überlegungen zu VPC-Endpunkten ... 1919
Verfügbarkeit in Regionen ... 1919
Erstellen eines VPC-Schnittstellen-Endpunkts für die Amazon DocumentDB-API ... 1920
Erstellen einer VPC-Endpunktrichtlinie für die Amazon DocumentDB-API ... 1921
Bewährte Methoden für die Gewährleistung der Sicherheit ... 1922
Prüfereignisse ... 1923
Unterstützte Ereignisse ... 1924
Auditing aktivieren ... 1930
Deaktivieren Sie die Überwachung ... 1938
Greifen Sie auf Ihre Veranstaltungen zu ... 1941
DML-Ereignisse filtern ... 1941
Amazon VPC und Amazon DocumentDB ... 1947
DocumentDB-Cluster in einer VPC ... 1947

Zugreifen auf einen Cluster in einer VPC ... 1961
Erstellen Sie eine IPv4-only VPC für einen Cluster ... 1963
Erstellen Sie eine Dual-Stack-VPC für einen Cluster ... 1971
Sichern und Wiederherstellen ... 1983
Sichern und Wiederherstellen: Konzepte ... 1984
Grundlegendes zur Sicherungsspeicher-Nutzung in ... 1987
Daten speichern, wiederherstellen, importieren und exportieren ... 1989
`Mongodump` ... 1989
`mongorestore` ... 1990
`Mongoexport` ... 1990
`Mongoimport` ... 1991
Tutorial ... 1992
Überlegungen zum Cluster-Snapshot ... 1994
Backup-Speicher ... 1995
Backup window ... 1996
Aufbewahrungszeitraum für Backups ... 1998
Cluster-Snapshot-Verschlüsselung kopieren ... 1998
Vergleich von automatischen und manuellen Snapshots ... 1999
Einen manuellen Cluster-Snapshot erstellen ... 2001
Einen Cluster-Snapshot kopieren ... 2004
Kopieren freigegebener Snapshots ... 2005
Schnappschüsse kopieren AWS-Regionen ... 2005
Einschränkungen ... 2005
Umgang mit Verschlüsselungen ... 2006
Überlegungen zu Parametergruppen ... 2006
Einen Cluster-Snapshot kopieren ... 2007
Freigeben eines Cluster-Snapshots ... 2017
Freigeben eines verschlüsselten Snapshots ... 2017
Freigeben eines Snapshots ... 2021
Wiederherstellung aus einem Cluster-Snapshot ... 2025
Wiederherstellung auf einen bestimmten Zeitpunkt ... 2032
Löschen eines Cluster-Snapshots ... 2039
Überwachung und Protokollierung ... 2042
Überwachung des Status eines Clusters ... 2044
Werte für den Cluster-Status ... 2044
Überwachung des Status eines Clusters ... 2046

Überwachung des Status einer Instanz ... 2047
Instance-Statuswerte ... 2048
Überwachen Sie den Instanzstatus mithilfe der AWS-Managementkonsole oder AWS CLI 2051
Werte für den Integritätsstatus der Instanz ... 2053
Überwachen Sie den Integritätsstatus der Instanz mit dem AWS-Managementkonsole ... 2053
Empfehlungen von Amazon DocumentDB anzeigen ... 2055
Ereignisabonnements ... 2058
Abonnieren von Ereignissen ... 2059
-Abonnements verwalten ... 2061
Kategorien und Nachrichten ... 2065
Überwachung von Amazon DocumentDB mit CloudWatch ... 2068
Amazon DocumentDB-Metriken ... 2068
CloudWatch Daten anzeigen ... 2089
Abmessungen von Amazon DocumentDB ... 2096
Überwachung der Opcounter-Metriken ... 2096
Überwachen von Datenbankverbindungen ... 2096
Protokollieren Amazon DocumentDB DocumentDB-API-Aufrufen mit CloudTrail ... 2097
Amazon DocumentDB DocumentDB-Informationen in CloudTrail ... 2097
Profilerstellung von Vorgängen ... 2098
Unterstützte Vorgänge ... 2099
Einschränkungen ... 2100
Den Profiler aktivieren ... 2100
Den Profiler deaktivieren ... 2105
Der Export von Profiler-Protokollen wird deaktiviert ... 2106
Zugriff auf Ihre Profiler-Logs ... 2108
Häufig gestellte Abfragen ... 2109
Überwachung mit Performance Insights ... 2109
Konzepte von Performance Insights ... 2111
Aktivieren und Deaktivieren von Performance Insights ... 2115
Konfigurieren von Zugriffsrichtlinien für Performance Insights ... 2118
Analyse der Metriken mit dem Performance-Insights-Dashboard ... 2122
Abrufen von Metriken mit der Performance-Insights-API ... 2143
CloudWatch Amazon-Metriken für Performance Insights ... 2158
Leistungseinblicke für Zählmetriken ... 2161
Serverless ... 2163
Anwendungsfälle für Serverless ... 2163

Verwendung von Amazon DocumentDB serverless für bestehende bereitgestellte Workloads ... 2165
Vorteile von Amazon DocumentDB serverless ... 2166
Wie funktioniert Serverless ... 2167
-Übersicht ... 2167
Konfigurationen für Amazon DocumentDB-Cluster ... 2169
Serverlose Skalierungskapazität von Amazon DocumentDB ... 2169
Serverlose Skalierung von Amazon DocumentDB ... 2171
Ruhezustand (0,5 DCUs) ... 2173
Anforderungen und Einschränkungen im Bereich Serverless ... 2174
Anforderungen an die Funktion ... 2174
Erstellen eines Clusters, das Serverless verwendet ... 2175
Einen serverlosen Amazon DocumentDB-Cluster erstellen ... 2176
Hinzufügen einer serverlosen Amazon DocumentDB-Instance ... 2179
Migration zu serverlos ... 2180
Migration vorhandener DocumentDB-Cluster zu serverlosen Clustern ... 2180
Migration von MongoDB zu DocumentDB serverless ... 2185
Serverlose Verwaltung ... 2186
Die Konfiguration des Skalierungskapazitätsbereichs eines Clusters anzeigen und ändern 2186
Die Angebotsstufe für serverlose Lesegeräte anzeigen und ändern ... 2189
Grenzwerte für serverlose Instanzen ... 2191
Serverlose Skalierungskonfiguration ... 2194
Auswahl des Skalierungskapazitätsbereichs für einen serverlosen DocumentDB-Cluster .. 2194
Auswahl der `MinCapacity` Einstellung für einen serverlosen DocumentDB-Cluster ... 2195
Auswahl der `MaxCapacity` Einstellung für einen serverlosen DocumentDB-Cluster ... 2197
Vermeiden von Fehlern aufgrund von zu wenig Speicher ... 2199
Warum wird meine serverlose Instance nicht herunterskaliert? ... 2201
Serverlose Überwachung ... 2202
Nicht genügend Arbeitsspeicher: Status inkompatibler Parameter ... 2202
Amazon-Metriken für serverlose DocumentDB CloudWatch ... 2202
Überwachung der serverlosen DocumentDB-Leistung mit Performance Insights ... 2207
Elastische Cluster ... 2209
Anwendungsfälle für elastische Cluster ... 2210
Benutzerprofile ... 2210
Inhaltsmanagement und historische Aufzeichnungen ... 2210
Vorteile elastischer Cluster ... 2210

AWS Serviceintegration ... 2211
Verfügbarkeit von Regionen und Versionen ... 2211
Regionale Verfügbarkeit für elastische Cluster ... 2211
Verfügbarkeit der Versionen ... 2212
Einschränkungen ... 2212
Elastische Clusterverwaltung ... 2212
Abfrage- und Schreibvorgänge ... 2213
Sammlungs- und Indexverwaltung ... 2213
Verwaltung und Diagnose ... 2214
Opt-in features ... 2214
Funktionsweise ... 2214
Elastisches Cluster-Sharding von Amazon DocumentDB ... 2215
Elastische Cluster-Migration ... 2219
Elastische Cluster-Skalierung ... 2219
Elastische Cluster-Zuverlässigkeit ... 2219
Elastischer Cluster-Speicher und Verfügbarkeit ... 2219
Funktionale Unterschiede zwischen Amazon DocumentDB 4.0 und Elastic Clusters ... 2220
Erste Schritte ... 2221
Voraussetzungen ... 2222
Schritt 1: Erstellen Sie einen elastischen Cluster ... 2223
Schritt 2: Stellen Sie eine Verbindung zu Ihrem Elastic Cluster her ... 2230
Schritt 3: Teilen Sie Ihre Sammlung, fügen Sie Daten ein und fragen Sie sie ab ... 2231
Schritt 4: Erkunden ... 2234
Bewährte Methoden ... 2235
Auswahl von Shard-Schlüsseln ... 2235
Verbindungsverwaltung ... 2236
Ungeteilte Sammlungen ... 2236
Skalierung elastischer Cluster ... 2236
Überwachung elastischer Cluster ... 2237
Verwaltung elastischer Cluster ... 2237
Elastische Cluster-Konfigurationen ändern ... 2238
Überwachung eines elastischen Clusters ... 2241
Löschen eines elastischen Clusters ... 2245
Verwaltung von Elastic Cluster-Snapshots ... 2247
Stoppen und Starten eines elastischen Clusters ... 2263
Aufrechterhaltung elastischer Cluster ... 2267

Datenverschlüsselung im Ruhezustand ... 2276
So verwenden Amazon DocumentDB Elastic Clusters Zuschüsse in AWS KMS ... 2278
Erstellen eines kundenseitig verwalteten Schlüssels ... 2279
Überwachung Ihrer Verschlüsselungsschlüssel für Amazon DocumentDB Elastic Clusters 2280
Weitere Informationen ... 2286
Service-linked Rollen ... 2286
Service-linked Rollenberechtigungen für elastische Cluster ... 2286
Kontingente ... 2291
Unterstützte Instance-Typen ... 2291
Unterstützte Regionen ... 2293
Regionale Kontingente ... 2295
Kontingente für die Aggregation ... 2298
Cluster-Kontingente ... 2298
Kontingente für Instanzen ... 2300
Benennungseinschränkungen: ... 2307
TTL-Einschränkungen ... 2309
Elastische Cluster-Kontingente ... 2310
Shard-Kontingente für elastische Cluster ... 2311
CPU-, Speicher-, Verbindungs- und Cursor-Kontingente pro Shard für Elastic Cluster ... 2311
Fehlerbehebung ... 2313
Probleme mit der Verbindung ... 2313
Es kann keine Verbindung zu einem Amazon DocumentDB-Endpunkt hergestellt werden . 2313
Testen einer Verbindung zu einer Amazon DocumentDB-Instance ... 2319
Verbindung zu einem ungültigen Endpunkt herstellen ... 2319
Die Treiberkonfiguration wirkt sich auf die Anzahl der Verbindungen aus ... 2320
Indizes ... 2320
Die Indexerstellung schlägt fehl ... 2321
Latenzprobleme und Fehlschläge bei der Erstellung des Hintergrundindexes ... 2321
Indexblähung ... 2322
Leistung und Ressourcenauslastung ... 2323
Statistiken zum Einfügen, Aktualisieren und Löschen anzeigen ... 2324
Analysieren Sie die Cache-Leistung ... 2325
Suchen und beenden Sie lang andauernde oder blockierte Abfragen ... 2327
Sehen Sie sich einen Abfrageplan an und optimieren Sie eine Abfrage ... 2328
Wie kann ich einen Abfrageplan in elastischen Clustern sehen? ... 2330
Listet alle laufenden Operationen auf einer Instanz auf ... 2333

Erfahren Sie, wann eine Abfrage Fortschritte macht ... 2335
Stellen Sie fest, warum ein System plötzlich langsam läuft ... 2338
Ermitteln der Ursache einer hohen CPU-Nutzung ... 2340
Finden Sie die offenen Cursor auf einer Instanz ... 2341
Die aktuelle Version der Amazon DocumentDB-Engine anzeigen ... 2341
Analysieren Sie die Indexnutzung und identifizieren Sie ungenutzte Indizes ... 2342
Identifizieren Sie fehlende Indizes ... 2344
Wie stelle ich fest, dass die Datenbanksammlung überfüllt ist? ... 2346
Zusammenfassung nützlicher Abfragen ... 2347
Müllabfuhr ... 2348
Grundlegendes zur Speicherbereinigung in Amazon DocumentDB ... 2349
Prozess der Müllabfuhr ... 2349
Speicherarchitektur und erweiterter Speicher ... 2350
Überwachung der Müllabfuhr ... 2351
Beispiel für eine CollStats-Ausgabe ... 2354
Häufig gestellte Fragen ... 2356
Fehlercodes ... 2359
Versionshinweise ... 2392
Versionshinweise zur Amazon DocumentDB-Engine ... 2393
Allgemeine Versionshinweise zu Amazon DocumentDB ... 2460
Versionshinweise zu Amazon DocumentDB Elastic Clusters ... 2465
Dokumentverlauf ... 2470
... mmcdlxxxiv

Was ist Amazon DocumentDB (mit MongoDB-Kompatibilität)

Amazon DocumentDB (mit MongoDB-Kompatibilität) ist ein schneller, zuverlässiger und vollständig verwalteter Datenbankdienst. Amazon DocumentDB macht es einfach, MongoDB-compatible Datenbanken in der Cloud einzurichten, zu betreiben und zu skalieren. Mit Amazon DocumentDB können Sie denselben Anwendungscode ausführen und dieselben Treiber und Tools verwenden, die Sie mit MongoDB verwenden.

Bevor Sie Amazon DocumentDB verwenden, sollten Sie sich mit den unter beschriebenen Konzepten und Funktionen vertraut machen. Funktionsweise Anschließend führen Sie die Schritte unter Leitfaden für die ersten Schritte aus.

Themen

- Überblick über Amazon DocumentDB
- Cluster
- Instances
- Regionen und Availability Zones
- Preise für Amazon DocumentDB
- Überwachen
- Schnittstellen
- Als nächstes
- Amazon DocumentDB: wie funktioniert es
- Was ist eine Dokumentendatenbank?

Überblick über Amazon DocumentDB

Im Folgenden sind einige allgemeine Funktionen von Amazon DocumentDB aufgeführt:

- Amazon DocumentDB unterstützt zwei Arten von Clustern: instanzbasierte Cluster und elastische Cluster. Elastische Cluster unterstützen Workloads mit Millionen von reads/writes pro Sekunde und Petabyte an Speicherkapazität. Weitere Informationen zu elastischen Clustern finden Sie unter. Verwenden von Amazon DocumentDB Elastic Clusters Der folgende Inhalt bezieht sich auf auf Amazon DocumentDB-Instance-basierte Cluster.

- Amazon DocumentDB vergrößert Ihr Speichervolumen automatisch, wenn Ihr Datenbankspeicherbedarf wächst. Ihr Speichervolumen wächst schrittweise um 10 GB, bis zu einem Maximum von 256 TiB für Engine-Version 8.0 und höher (128 TiB für frühere Engine-Versionen). Sie müssen in Hinblick auf zukünftiges Wachstum keinen zusätzlichen Speicher für Ihren Cluster bereitstellen.
- Mit Amazon DocumentDB können Sie den Lesedurchsatz erhöhen, um umfangreiche Anwendungsanforderungen zu unterstützen, indem Sie bis zu 15 Replikat-Instances erstellen. Amazon DocumentDB-Replikate nutzen denselben zugrundeliegenden Speicher, wodurch die Kosten gesenkt werden und Schreibvorgänge an den Replikatknoten vermieden werden müssen. Diese Funktion gibt mehr Rechenleistung für die Bearbeitung von Leseanforderungen frei und reduziert die Verzögerungen bei Replikaten — oft auf einstellige Millisekunden. Sie können Replikate unabhängig von der Größe des Speichervolumes innerhalb von Minuten hinzufügen. Amazon DocumentDB bietet auch einen Reader-Endpunkt, sodass die Anwendung eine Verbindung herstellen kann, ohne dass Replikate beim Hinzufügen und Entfernen nachverfolgt werden müssen.
- Mit Amazon DocumentDB können Sie die Rechen- und Speicherressourcen für jede Ihrer Instances nach oben oder unten skalieren. Skalierungsvorgänge bei der Datenverarbeitung dauern in der Regel nur wenige Minuten.
- Amazon DocumentDB wird in Amazon Virtual Private Cloud (Amazon VPC) ausgeführt, sodass Sie Ihre Datenbank in Ihrem eigenen virtuellen Netzwerk isolieren können. Sie können auch Firewalleinstellungen so konfigurieren, dass der Netzwerkzugriff auf Ihren Cluster gesteuert wird.
- Amazon DocumentDB überwacht kontinuierlich den Zustand Ihres Clusters. Bei einem Instance-Ausfall startet Amazon DocumentDB die Instance und die zugehörigen Prozesse automatisch neu. Amazon DocumentDB erfordert keine Wiederholung der Datenbank-Redo-Logs bei der Wiederherstellung nach einem Absturz, wodurch die Neustartzeiten erheblich reduziert werden. Amazon DocumentDB isoliert auch den Datenbank-Cache vom Datenbankprozess, sodass der Cache einen Instance-Neustart übersteht.
- Bei einem Instance-Ausfall automatisiert Amazon DocumentDB den Failover auf eines von bis zu 15 Amazon DocumentDB-Replikaten, die Sie in anderen Availability Zones erstellen. Wenn keine Replikate bereitgestellt wurden und ein Fehler auftritt, versucht Amazon DocumentDB, automatisch eine neue Amazon DocumentDB-Instance zu erstellen.
- Die Backup-Funktion in Amazon DocumentDB ermöglicht eine Point-in-Time-Wiederherstellung für Ihren Cluster. Diese Funktion ermöglicht Ihnen, Ihren Cluster zu jeder Sekunde innerhalb der Aufbewahrungsfrist bis zu den letzten 5 Minuten wiederherzustellen. Sie können den Aufbewahrungszeitraum für automatische Backups auf maximal 35 Tage festlegen. Automatisierte

Backups werden im Amazon Simple Storage Service (Amazon S3) gespeichert, der auf eine Beständigkeit von 99,999999999% ausgelegt ist. Amazon DocumentDB-Backups erfolgen automatisch, inkrementell und kontinuierlich und haben keine Auswirkungen auf die Leistung Ihres Clusters.

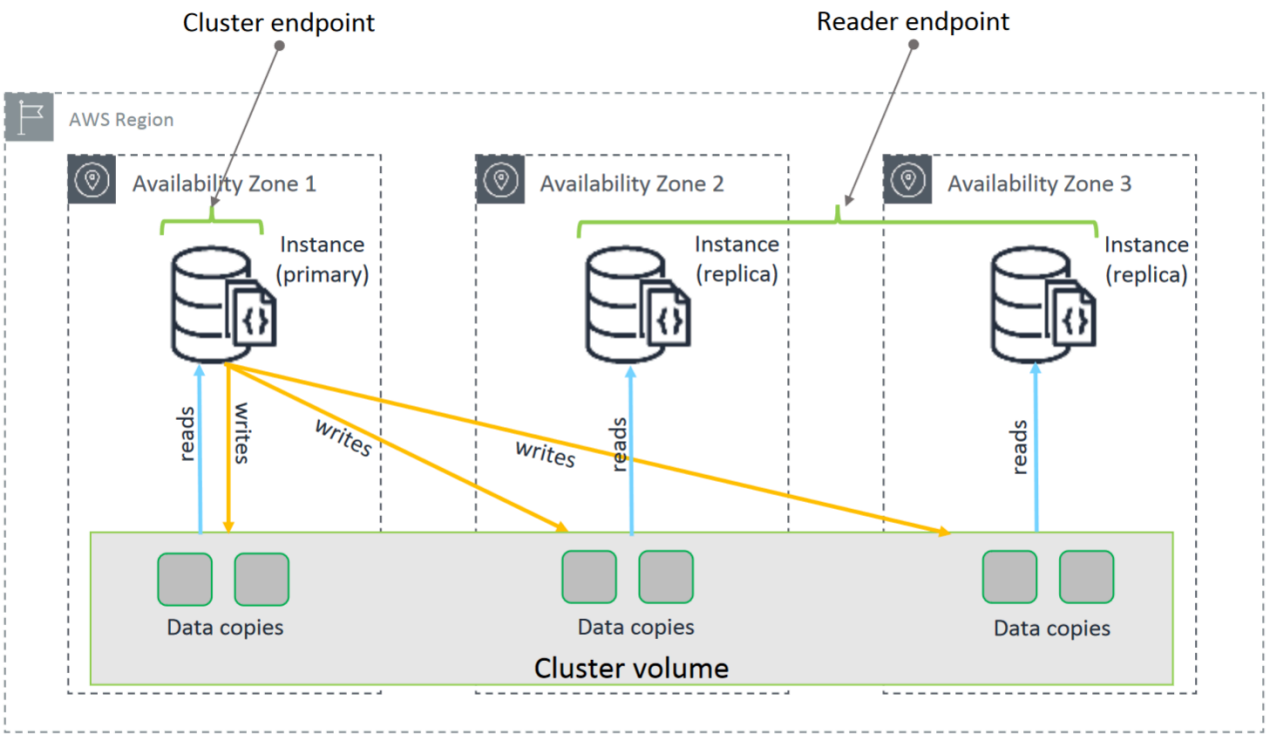
- Mit Amazon DocumentDB können Sie Ihre Datenbanken mit Schlüsseln verschlüsseln, die Sie über () erstellen und kontrollieren. AWS Key Management Service AWS KMS Auf einem Datenbank-Cluster, der mit Amazon DocumentDB-Verschlüsselung ausgeführt wird, werden Daten, die im zugrunde liegenden Speicher gespeichert sind, verschlüsselt. Die automatischen Sicherungen, Snapshots und Replicas im gleichen Cluster werden ebenfalls verschlüsselt.
- Amazon DocumentDB ist im Rahmen des Federal Risk and Authorization Management Program (FedRAMP) autorisiert. Es verfügt über die FedRAMP High-Autorisierung für AWS GovCloud (US-) Regionen und die FedRAMP Moderate-Autorisierung für US-Regionen. AWS East/West Einzelheiten zu den Maßnahmen zur AWS Einhaltung der Vorschriften finden Sie unter AWS Services in Scope by Compliance Program.

Wenn AWS Services für Sie neu sind, nutzen Sie die folgenden Ressourcen, um mehr zu erfahren:

- AWS bietet Dienste für Computer, Datenbanken, Speicher, Analysen und andere Funktionen. Eine Übersicht über alle AWS Dienste finden Sie unter Cloud Computing mit Amazon Web Services.
- AWS bietet eine Reihe von Datenbankdiensten. Hinweise dazu, welcher Dienst für Ihre Umgebung am besten geeignet ist, finden Sie unter Datenbanken auf AWS.

Cluster

Ein Cluster besteht aus 0 bis 16 Instanzen und einem Cluster-Speicher-Volume, das die Daten für diese Instances verwaltet. Alle Schreibvorgänge erfolgen über die primäre Instance. Alle Instances (primäre und Replicas) unterstützen Lesevorgänge. Die Daten des Clusters werden im Cluster-Volume gespeichert, mit Kopien in drei verschiedenen Availability Zones.



Instance-basierte Amazon DocumentDB 5.0-Cluster unterstützen zwei Speicherkonfigurationen für ein Datenbank-Cluster: Amazon DocumentDB Standard und Amazon DocumentDB. I/O-optimized Weitere Informationen finden Sie unter Amazon DocumentDB-Cluster-Speicherkonfigurationen.

Instances

Eine Amazon DocumentDB-Instance ist eine isolierte Datenbankumgebung in der Cloud. Eine Instance kann mehrere von Benutzern erstellte Datenbanken enthalten. Sie können eine Instanz mithilfe von AWS-Managementkonsole oder erstellen und ändern. AWS CLI

Die Rechen- und Speicherkapazität einer Instanz werden durch ihre Instanzklasse bestimmt. Sie können die Instance auswählen, die Ihren Anforderungen am besten entspricht. Wenn sich Ihre Anforderungen im Laufe der Zeit ändern, können Sie eine andere Instance-Klasse wählen. Spezifikationen für DB-Instance-Klassen finden Sie unter Spezifikationen der Instanzklasse

Amazon DocumentDB-Instances werden nur in der Amazon VPC-Umgebung ausgeführt. Amazon VPC gibt Ihnen die Kontrolle über Ihre virtuelle Netzwerkumgebung: Sie können Ihren eigenen IP-Adressbereich wählen, Subnetze erstellen und Routing- und Zugriffskontrolllisten (ACLs) konfigurieren.

Bevor Sie Amazon DocumentDB-Instances erstellen können, müssen Sie einen Cluster erstellen, der die Instances enthält.

Nicht alle Instanzklassen werden in allen AWS-Region unterstützt. Die folgende Tabelle zeigt, welche Instanzklassen in den einzelnen Regionen unterstützt werden.

> **Note**
>
> Eine vollständige Liste der Instance-Typen, die von Amazon DocumentDB in jeder Instance-Klasse unterstützt werden, finden Sie unterSpezifikationen der Instanzklasse.

Unterstützte Instance-Klassen nach Region

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Serverless
USA Ost (Ohio)	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt
USA Ost (Nord-Virginia)	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt
USA West (Oregon)	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt
Afrika (Kapstadt)			Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Südamerika (São Paulo)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Asien-Pazifik (Hongkong)	Unterstützt		Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Asien-Pazifik (Hyderabad)			Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Asien-Pazifik (Malaysia)			Unterstützt			Unterstützt	Unterstützt	Unterstützt

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Serverless
Asien-Pazifik (Mumbai)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Asien-Pazifik (Osaka)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Asien-Pazifik (Seoul)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Asien-Pazifik (Sydney)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Asien-Pazifik (Jakarta)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Asien-Pazifik (Melbourne)			Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Asien-Pazifik (Singapur)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Asien-Pazifik (Thailand)			Unterstützt			Unterstützt	Unterstützt	Unterstützt
Asien-Pazifik (Tokio)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Kanada (Zentral)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Kanada West (Calgary)			Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Europa (Frankfurt)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Serverless
Europa (Zürich)		Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Europa (Irland)	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt
Europa (London)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Europa (Milan)			Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Europa (Paris)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Europa (Spain)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Europa (Stockholm)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Mexiko (Zentral)			Unterstützt			Unterstützt	Unterstützt	Unterstützt
Naher Osten (VAE)			Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
China (Peking)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
China (Ningxia)	Unterstützt		Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Israel (Tel Aviv)			Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Serverless
AWS GovCloud (US-West)	Unterstützt	Unterstützt	Unterstützt	Unterstützt			Unterstützt	Unterstützt
AWS GovCloud (US-East)		Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt

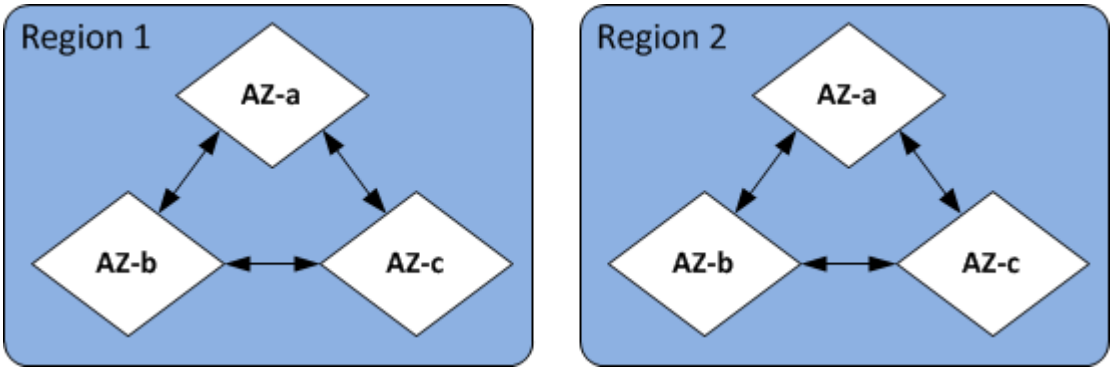
Regionen und Availability Zones

AWS-Regionen und Availability Zones definieren die physischen Standorte Ihres Clusters und Ihrer Instances.

Regionen

AWS Cloud-Computing-Ressourcen befinden sich in hochverfügbaren Rechenzentren in verschiedenen Regionen der Welt (z. B. Nordamerika, Europa oder Asien). Jeder Rechenzentrumsstandort wird als AWS-Region bezeichnet.

Jedes AWS-Region ist so konzipiert, dass es vollständig vom anderen isoliert ist AWS-Regionen. Innerhalb jeder gibt es mehrere Availability Zones (Verfügbarkeitszonen). Durch das Starten Ihrer Knoten in verschiedenen Availability Zones können Sie eine größtmögliche Fehlertoleranz zu erreichen. Das folgende Diagramm zeigt einen allgemeinen Überblick über die Funktionsweise AWS-Regionen von Availability Zones.



Availability Zones

Jeder AWS-Region enthält mehrere unterschiedliche Standorte, die als Availability Zones bezeichnet werden. Jede Availability Zone wurde so konzipiert, dass sie von Fehlern in anderen

Availability Zones isoliert ist und eine kostengünstige Netzwerkverbindung mit geringer Latenz zu anderen Availability Zones in derselben Region bereitstellt. Indem Instances für einen bestimmten Cluster in mehreren Availability Zones gestartet werden, können Sie Ihre Anwendungen vor dem unwahrscheinlichen Fall des Fehlschlagens einer Availability Zone schützen.

Die Amazon DocumentDB-Architektur trennt Speicher und Rechenleistung. Für die Speicherebene repliziert Amazon DocumentDB sechs Kopien Ihrer Daten in drei AWS Availability Zones. Wenn Sie beispielsweise einen Amazon DocumentDB-Cluster in einer Region starten, die nur zwei Availability Zones unterstützt, wird Ihr Datenspeicher sechsmal über drei Availability Zones repliziert, Ihre Recheninstanzen sind jedoch nur in zwei Availability Zones verfügbar.

In der folgenden Tabelle ist die Anzahl der Availability Zones aufgeführt, die Sie in einer bestimmten Umgebung verwenden können AWS-Region , um Recheninstanzen für Ihren Cluster bereitzustellen.

Name der Region	Region	Availability Zones (Berechnung)
USA Ost (Ohio)	`us-east-2`	3
USA Ost (Nord-Virginia)	`us-east-1`	6
USA West (Oregon)	`us-west-2`	4
Afrika (Kapstadt)	`af-south-1`	3
Südamerika (São Paulo)	`sa-east-1`	3
Asien-Pazifik (Hongkong)	`ap-east-1`	3
Asien-Pazifik (Hyderabad)	`ap-south-2`	3
Asien-Pazifik (Malaysia)	`ap-southeast-5`	3
Asien-Pazifik (Mumbai)	`ap-south-1`	3

Name der Region	Region	Availability Zones (Berechnung)
Asien-Pazifik (Osaka)	ap-northeast-3	3
Asien-Pazifik (Seoul)	ap-northeast-2	4
Asien-Pazifik (Singapur)	ap-southeast-1	3
Asien-Pazifik (Sydney)	ap-southeast-2	3
Asien-Pazifik (Jakarta)	ap-southeast-3	3
Asien-Pazifik (Melbourne)	ap-southeast-4	3
Asien-Pazifik (Thailand)	ap-southeast-7	3
Asien-Pazifik (Tokio)	ap-northeast-1	3
Kanada (Zentral)	ca-central-1	3
Kanada West (Calgary)	ca-west-1	3
Region China (Peking)	cn-north-1	3
China (Ningxia)	cn-northwest-1	3
Europa (Frankfurt)	eu-central-1	3
Europa (Zürich)	eu-central-2	3
Europa (Irland)	eu-west-1	3
Europa (London)	eu-west-2	3

Name der Region	Region	Availability Zones (Berechnung)
Europa (Milan)	`eu-south-1`	3
Europa (Paris)	`eu-west-3`	3
Europa (Spain)	`eu-south-2`	3
Europa (Stockholm)	`eu-north-1`	3
Mexiko (Zentral)	`mx-central-1`	3
Naher Osten (VAE)	`me-central-1`	3
Israel (Tel Aviv)	`il-central-1`	3
AWS GovCloud (US-West)	`us-gov-west-1`	3
AWS GovCloud (US-East)	`us-gov-east-1`	3

Preise für Amazon DocumentDB

Amazon DocumentDB-Cluster werden auf der Grundlage der folgenden Komponenten abgerechnet:

- Instance-Stunden (pro Stunde) — Basierend auf der Instance-Klasse der Instance (z. B.). `db.r5.xlarge` Die Preise werden auf Stundenbasis aufgeführt, aber Rechnungen werden jetzt auf die Sekunde genau kalkuliert und zeigen die Zeiten im Dezimalformat an. Die Nutzung von Amazon DocumentDB wird in Sekundenschritten mit einem Mindestzeitraum von 10 Minuten in Rechnung gestellt. Weitere Informationen finden Sie unter Instanzklassen verwalten.
- I/O Anfragen (pro 1 Million Anfragen pro Monat) — Gesamtzahl der I/O Speicheranforderungen, die Sie in einem Abrechnungszeitraum stellen.
- Backup-Speicher (pro GiB pro Monat) — Backup-Speicher ist der Speicher, der automatisierten Datenbanksicherungen und allen aktiven Datenbank-Snapshots, die Sie erstellt haben, zugeordnet ist. Wenn Sie die Aufbewahrungszeit Ihrer Backups erhöhen oder zusätzliche Datenbank-Snapshots erstellen, belegt Ihre Datenbank dementsprechend mehr Backup-Speicher. Der Backup-

Speicher wird dosiert GB-months und gilt nicht pro Sekunde. Weitere Informationen finden Sie unter Sichern und Wiederherstellen in Amazon DocumentDB.

- Datenübertragung (pro GB) — Datenübertragung in und aus Ihrer Instance vom oder zum Internet oder aus anderen AWS-Regionen Quellen.

Detaillierte Informationen finden Sie unter Amazon DocumentDB-Preise.

Kostenlose Testversion

Mit der einmonatigen kostenlosen Testversion können Sie Amazon DocumentDB kostenlos testen. Weitere Informationen finden Sie unter Kostenlose Testversion auf der Preisseite von Amazon DocumentDB oder in den häufig gestellten Fragen zur kostenlosen Amazon DocumentDB-Testversion.

Überwachen

Es gibt verschiedene Möglichkeiten, die Leistung und den Zustand einer Instance zu überwachen. Sie können den kostenlosen CloudWatch Amazon-Service verwenden, um die Leistung und den Zustand einer Instance zu überwachen. Leistungsdiagramme finden Sie auf der Amazon DocumentDB-Konsole. Sie können Amazon DocumentDB-Ereignisse abonnieren, um benachrichtigt zu werden, wenn Änderungen an einer Instance, einem Snapshot, einer Parametergruppe oder einer Sicherheitsgruppe vorgenommen werden.

Weitere Informationen finden Sie hier:

- Überwachung von Amazon DocumentDB mit CloudWatch
- Protokollieren Amazon DocumentDB DocumentDB-API-Aufrufen mit AWS CloudTrail

Schnittstellen

Es gibt mehrere Möglichkeiten für Sie, mit Amazon DocumentDB zu interagieren, darunter die AWS-Managementkonsole und die. AWS CLI

AWS-Managementkonsole

Das AWS-Managementkonsole ist eine einfache webbasierte Benutzeroberfläche. Sie können Ihre Cluster und Instances von der Konsole aus verwalten, ohne dass eine Programmierung erforderlich ist. Um auf die Amazon DocumentDB-Konsole zuzugreifen, melden Sie sich bei der

an AWS-Managementkonsole und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb

AWS CLI

Sie können die AWS Command Line Interface (AWS CLI) verwenden, um Ihre Amazon DocumentDB-Cluster und Instances zu verwalten. Mit einer minimalen Konfiguration können Sie beginnen, alle Funktionen der Amazon DocumentDB-Konsole von Ihrem bevorzugten Terminalprogramm aus zu nutzen.

- Informationen zur AWS CLI Installation finden Sie unter Installation der AWS Befehlszeilenschnittstelle.
- Informationen zum Einstieg in die Verwendung von AWS CLI für Amazon DocumentDB finden Sie unter Referenz zur AWS Befehlszeilenschnittstelle für Amazon DocumentDB.

MongoDB-Treiber

Zum Entwickeln und Schreiben von Anwendungen für einen Amazon DocumentDB-Cluster können Sie die MongoDB-Treiber auch mit Amazon DocumentDB verwenden. Weitere Informationen finden Sie auf der Registerkarte MongoDB-Shell in oder. Verbindung bei aktiviertem TLS herstellen Verbindung bei deaktiviertem TLS herstellen

Als nächstes

In den vorangegangenen Abschnitten wurden Ihnen die grundlegenden Infrastrukturkomponenten vorgestellt, die Amazon DocumentDB bietet. Was sollten Sie als nächstes tun? Je nach Ihren Umständen finden Sie einen der folgenden Themen, um loszulegen:

- Beginnen Sie mit Amazon DocumentDB, indem Sie einen Cluster und eine Instance mithilfe CloudFormation Amazon DocumentDB: Schnellstart mit CloudFormation von Amazon DocumentDB erstellen.
- Beginnen Sie mit Amazon DocumentDB, indem Sie einen Cluster und eine Instance mithilfe der Anweisungen in unserer erstellen. Leitfaden für die ersten Schritte
- Beginnen Sie mit Amazon DocumentDB, indem Sie anhand der Anweisungen unter einen elastischen Cluster erstellen. Erste Schritte mit Amazon DocumentDB Elastic Clusters
- Migrieren Sie Ihre MongoDB-Implementierung zu Amazon DocumentDB anhand der Anleitungen unter Migration und Aktualisierung von Amazon DocumentDB

Amazon DocumentDB: wie funktioniert es

Amazon DocumentDB (mit MongoDB-Kompatibilität) ist ein vollständig verwalteter MongoDB-compatible Datenbankdienst. Mit Amazon DocumentDB können Sie denselben Anwendungscode ausführen und dieselben Treiber und Tools verwenden, die Sie mit MongoDB verwenden. Amazon DocumentDB ist mit MongoDB 3.6, 4.0, 5.0 und 8.0 kompatibel.

Themen

- Amazon DocumentDB-Versionen
- Amazon DocumentDB-Endpunkte
- TLS-Unterstützung
- Amazon DocumentDB-Speicher
- Amazon DocumentDB-Replikation
- Zuverlässigkeit von Amazon DocumentDB
- Präferenzoptionen lesen
- TTL löscht
- Abrechnungsfähige Ressourcen

Amazon DocumentDB-Versionen

> **Note**
>
> Die Nummerierung von Nebenversionen gilt für Amazon DocumentDB 5.0 und höher. Amazon DocumentDB 3.6 und 4.0 verwenden nur Hauptversionen mit Engine-Patches.

Ab Amazon DocumentDB 5.0 verwendet die Engine-Version ein vierstufiges Nummerierungsschema: major.major.minor.patch.

- Hauptversion — Identifiziert durch die ersten beiden Nummern der Version (z. B. 5.0). Die Hauptversion steht für den MongoDB-Kompatibilitätsgrad. Hauptversionen führen neue Funktionen, Leistungsverbesserungen und MongoDB-Kompatibilitätsupdates ein. Für das Upgrade zwischen Hauptversionen (z. B. von 4.0 auf 5.0) ist ein Hauptversions-Upgrade (MVU) erforderlich. Weitere Informationen finden Sie unter Direktes Upgrade der Hauptversion von Amazon DocumentDB.

- Nebenversion — Identifiziert durch die dritte Versionsnummer (z. B. 1 in 5.0.1). Nebenversionen enthalten Verbesserungen wie neue Funktionen, Sicherheitsupdates und Bugfixes innerhalb derselben Hauptversion. Sie können manuell auf eine neue Nebenversion aktualisieren. Weitere Informationen finden Sie unter Upgrade der Nebenversion von Amazon DocumentDB.
- Patch-Version — Engine-Patches werden innerhalb einer Nebenversion angewendet und enthalten wichtige Sicherheits- und Bugfixes. Die Patch-Version wird anhand der Engine-Patch-Version (z. B.`3.0.17983`) nachverfolgt, die Sie überprüfen können, indem Sie Folgendes ausführen`db.runCommand({getEngineVersion: 1})`. Patches werden während des Wartungsfensters Ihres Clusters angewendet. Weitere Informationen finden Sie unter Wartung von Amazon DocumentDB.

Wenn Sie einen neuen Cluster erstellen, können Sie jede verfügbare Engine-Version angeben. Informationen zu verfügbaren Versionen und Supportdaten finden Sie unterUnterstützungstermine der Amazon DocumentDB DocumentDB-Engine-Version.

Wenn Sie Amazon DocumentDB verwenden, erstellen Sie zunächst einen Cluster. Ein DB-Cluster besteht aus null oder mehreren Datenbank-Instances und einem Cluster-Volume, das die Daten für diese Instances verwaltet. Ein Amazon DocumentDB-Cluster-Volume ist ein virtuelles Datenbank-Speicher-Volume, das sich über mehrere Availability Zones erstreckt. Jede Availability Zone verfügt über eine Kopie der Cluster-Daten.

Ein Amazon DocumentDB-Cluster besteht aus zwei Komponenten:

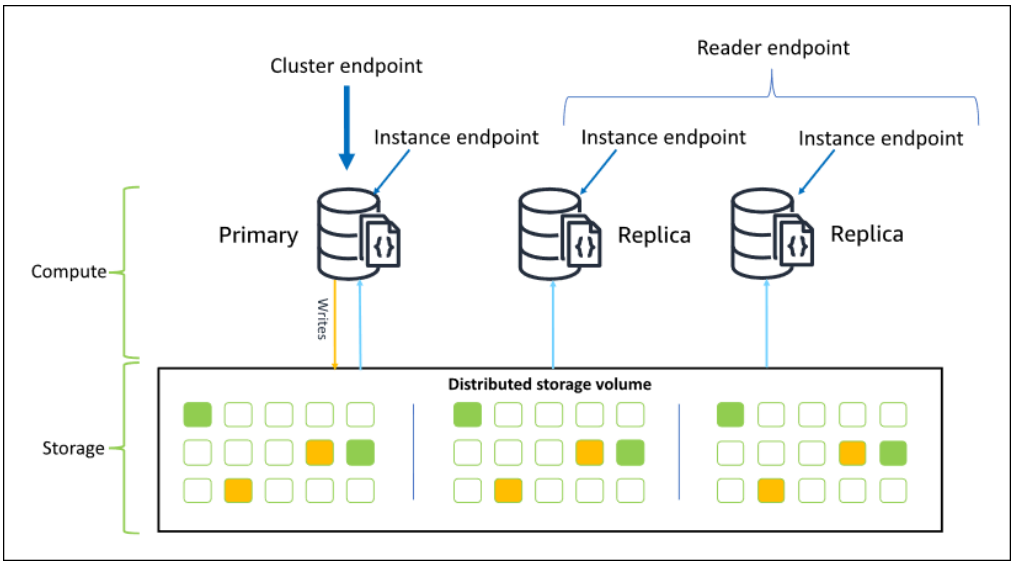
- Cluster-Volume — Verwendet einen Cloud-nativen Speicherservice, um Daten auf sechs Arten über drei Availability Zones zu replizieren und so einen äußerst langlebigen und verfügbaren Speicher bereitzustellen. Ein Amazon DocumentDB-Cluster hat genau ein Cluster-Volume, auf dem bis zu 256 TiB an Daten für Engine-Version 8.0 und höher gespeichert werden können (128 TiB für frühere Engine-Versionen).
- Instances — Stellen die Rechenleistung für die Datenbank bereit und schreiben Daten auf das Cluster-Speicher-Volume und lesen Daten von diesem. Ein Amazon DocumentDB-Cluster kann 0—16 Instances haben.

Instances erfüllen eine von zwei Rollen:

- Primäre Instanz — Unterstützt Lese- und Schreibvorgänge und führt alle Datenänderungen am Cluster-Volume durch. Jeder Amazon DocumentDB-Cluster hat eine primäre Instance.

- Replica-Instance — Unterstützt nur Lesevorgänge. Ein Amazon DocumentDB-Cluster kann zusätzlich zur primären Instance bis zu 15 Replikate haben. Die Verwendung mehrerer Replikate ermöglicht es Ihnen, die Leseauslastungen zu verteilen. Darüber hinaus erhöhen Sie durch die Platzierung von Replikaten in separaten Availability Zones auch die Cluster-Verfügbarkeit.

Das folgende Diagramm veranschaulicht die Beziehung zwischen dem Cluster-Volume, der primären Instance und den Replikaten in einem Amazon DocumentDB-Cluster:



Cluster-Instances müssen nicht von derselben Instance-Klasse sein. Sie können beliebig bereitgestellt und beendet werden. Mit dieser Architektur können Sie die Rechenkapazität Ihres Clusters unabhängig von seiner Storage-Funktionalität skalieren.

Wenn Ihre Anwendung Daten in die Primär-Instance schreibt, schreibt diese die Daten dauerhaft in das Cluster-Volume. Anschließend repliziert es den Status dieses Schreibvorgangs (nicht die Daten) auf jedes aktive Replikat. Amazon DocumentDB-Replikate sind nicht an der Verarbeitung von Schreibvorgängen beteiligt, weshalb Amazon DocumentDB-Replikate für die Skalierung von Lesevorgängen von Vorteil sind. Lesevorgänge aus Amazon DocumentDB-Replikaten sind letztlich konsistent, wobei die Replikatverzögerung minimal ist — in der Regel weniger als 100 Millisekunden, nachdem die primäre Instance die Daten geschrieben hat. Lesezugriffe von den Replikaten werden garantiert in der Reihenfolge gelesen, in der sie auf die primäre Instance geschrieben wurden. Die Replikationsverzögerung hängt von der Rate der Datenänderung ab. Perioden mit hoher Schreibaktivität können die Replikationsverzögerung erhöhen. Weitere Informationen finden Sie in den `ReplicationLag`-Metriken unter Amazon DocumentDB-Metriken.

Amazon DocumentDB-Endpunkte

Amazon DocumentDB bietet mehrere Verbindungsoptionen für eine Vielzahl von Anwendungsfällen. Um eine Verbindung zu einer Instance in einem Amazon DocumentDB-Cluster herzustellen, geben Sie den Endpunkt der Instance an. Ein Endpunkt ist eine Host-Adresse und eine Portnummer, getrennt durch einen Doppelpunkt.

Stellen Sie mithilfe des Cluster-Endpunkts im Replikat-Set-Modus eine Verbindung zu Ihrem Cluster her (siehe Verbindung zu Amazon DocumentDB als Replikatsatz herstellen). Verwenden Sie diesen Ansatz, es sei denn, Sie müssen eine Verbindung zum Reader-Endpunkt oder einem Instance-Endpunkt herstellen. Um Anforderungen an Ihre Replikate weiterzuleiten, wählen Sie eine Treibereinstellung für die Leseeinstellung aus, die die Leseskalierung maximiert und gleichzeitig die Anforderungen für die Lesekonsistenz Ihrer Anwendung erfüllt. Die Leseeinstellung `secondaryPreferred` ermöglicht Replica-Lesevorgänge, sodass die primäre Instance produktiver sein kann.

Die folgenden Endpunkte sind in einem Amazon DocumentDB-Cluster verfügbar.

Cluster-Endpunkt

Der Cluster-Endpunkt verbindet sich mit der aktuellen primären Instance Ihres Clusters. Der Cluster-Endpunkt kann für Lese- und Schreibvorgänge verwendet werden. Ein Amazon DocumentDB-Cluster hat genau einen Cluster-Endpunkt.

Der Cluster-Endpunkt bietet Failover-Support für Lese-/Schreibverbindungen zum Cluster. Wenn die aktuelle primäre Instance Ihres Clusters ausfällt und Ihr Cluster mindestens eine aktive Read Replica hat, leitet der Cluster-Endpunkt Verbindungsanforderungen automatisch an eine neue primäre Instance weiter. Wenn Sie eine Verbindung zu Ihrem Amazon DocumentDB-Cluster herstellen, verwenden Sie den Cluster-Endpunkt im Replikat-Set-Modus (siehe). Verbindung zu Amazon DocumentDB als Replikatsatz herstellen

Im Folgenden finden Sie ein Beispiel für einen Amazon DocumentDB-Cluster-Endpunkt:

```
sample-cluster.cluster-123456789012.us-east-1.docdb.amazonaws.com:27017
```

Im Folgenden finden Sie ein Beispiel für eine Verbindungszeichenfolge für diesen Cluster-Endpunkt:

```
mongodb://username:password@sample-cluster.cluster-123456789012.us-
east-1.docdb.amazonaws.com:27017
```

Informationen zum Suchen der Endpunkte eines Clusters finden Sie unter Die Endpunkte eines Clusters finden.

Leser-Endpunkt

Der Reader-Endpunkt agiert als Load-Balancer für schreibgeschützte Verbindungen für alle verfügbaren Replikate in Ihrem Cluster. Ein Cluster-Reader-Endpunkt fungiert als Cluster-Endpunkt, wenn Sie eine Verbindung über den `replicaSet` Modus herstellen. Das heißt, in der Verbindungszeichenfolge lautet der Parameter für das Replikatset. `&replicaSet=rs0` In diesem Fall können Sie Schreibvorgänge auf dem Primärserver ausführen. Wenn Sie jedoch eine Verbindung zum Cluster herstellen, indem Sie angeben`directConnection=true`, dass der Versuch, einen Schreibvorgang über eine Verbindung zum Reader-Endpunkt durchzuführen, zu einem Fehler führt. Ein Amazon DocumentDB-Cluster hat genau einen Reader-Endpunkt.

Wenn der Cluster nur eine (primäre) Instance enthält, verbindet sich der Reader-Endpunkt mit der primären Instance. Wenn Sie Ihrem Amazon DocumentDB-Cluster eine Replica-Instance hinzufügen, öffnet der Reader-Endpunkt schreibgeschützte Verbindungen zum neuen Replikat, nachdem es aktiv ist.

Im Folgenden finden Sie ein Beispiel für einen Reader-Endpunkt für einen Amazon DocumentDB-Cluster:

```
sample-cluster.cluster-ro-123456789012.us-east-1.docdb.amazonaws.com:27017
```

Im Folgenden finden Sie ein Beispiel für eine Verbindungszeichenfolge unter Verwendung eines Reader-Endpunkts:

```
mongodb://username:password@sample-cluster.cluster-ro-123456789012.us-
east-1.docdb.amazonaws.com:27017
```

Der Reader-Endpunkt verteilt nur die Last der Read-only-Verbindungen – nicht die der Leseanforderungen. Wenn einige Reader-Endpunktverbindungen stärker genutzt werden als andere, sind Ihre Leseanforderungen möglicherweise nicht gleichmäßig zwischen Instances im Cluster verteilt. Es wird empfohlen, dass Sie zum Verteilen von Anforderungen eine Verbindung zum Clusterendpunkt als Replikatsatz herstellen und die Lesevorstellungsoption secondaryPreferred nutzen.

Informationen zum Suchen der Endpunkte eines Clusters finden Sie unter Die Endpunkte eines Clusters finden.

Instance-Endpunkt

Ein Instance-Endpunkt verbindet sich mit einer bestimmten Instance innerhalb Ihres Clusters. Der Instance-Endpunkt für die aktuelle Primär-Instance kann für Lese- und Schreibvorgänge verwendet werden. Der Versuch, Schreiboperationen auf einen Instance-Endpunkt für ein Lesereplikat durchzuführen, führt jedoch zu einem Fehler. Ein Amazon DocumentDB-Cluster hat einen Instance-Endpunkt pro aktiver Instance.

Ein Instance-Endpunkt bietet für Szenarien, in denen der Cluster-Endpunkt oder der Lese-Endpunkt möglicherweise nicht geeignet ist, direkte Kontrolle über Verbindungen zu einer bestimmten Instance. Ein Beispiel für einen Anwendungsfall ist die Bereitstellung für einen periodischen Read-Only-Analyse-Workload. Sie können eine Replikat-Instance bereitstellen, die größer als üblich ist, sich direkt mit der neuen größeren Instance und ihrem Instance-Endpunkt verbinden, die Analyseabfragen ausführen und die Instance dann beenden. Die Verwendung des Instance-Endpunkts verhindert, dass sich der Analyseverkehr auf andere Cluster-Instances auswirkt.

Im Folgenden finden Sie ein Beispiel für einen Instance-Endpunkt für eine einzelne Instance in einem Amazon DocumentDB-Cluster:

```
sample-instance.123456789012.us-east-1.docdb.amazonaws.com:27017
```

Im Folgenden finden Sie ein Beispiel für eine Verbindungszeichenfolge mit diesem Instance-Endpunkt:

```
mongodb://username:password@sample-instance.123456789012.us-
east-1.docdb.amazonaws.com:27017
```

> **Note**
>
> Die Rolle einer Instance als "Primär" oder "Replikat" kann sich aufgrund eines Failover-Ereignisses ändern. Ihre Anwendungen sollten niemals davon ausgehen, dass ein bestimmter Instance-Endpunkt die primäre Instance ist. Stellen Sie keine Verbindung zu Instance-Endpunkten für Produktionsanwendungen her. Verwenden Sie stattdessen den Cluster-Endpunkt im Replikatsatzmodus (siehe Verbindung zu Amazon DocumentDB als Replikatsatz herstellen). Weitere Informationen zur erweiterten Kontrolle der Instance-Failover-Priorität finden Sie unter Grundlegendes zur Amazon DocumentDB-Cluster-Fehlertoleranz.

Informationen zum Suchen der Endpunkte eines Clusters finden Sie unter Den Endpunkt einer Instanz finden.

Replikat-Set-Modus

Sie können im Replikatsatzmodus eine Verbindung zu Ihrem Amazon DocumentDB-Cluster-Endpunkt herstellen, indem Sie den Namen des Replikatsatzes angeben. `rs0` Die Verbindung im Replikatsatzmodus bietet die Möglichkeit, die Optionen Read Concern, Write Concern und Read Preference festzulegen. Weitere Informationen finden Sie unter Lesekonsistenz.

Im Folgenden finden Sie ein Beispiel für eine Verbindungszeichenfolge, die im Replikatsatzmodus verbunden ist:

```
mongodb://username:password@sample-cluster.cluster-123456789012.us-
east-1.docdb.amazonaws.com:27017/?replicaSet=rs0
```

Wenn Sie im Replica-Set-Modus eine Verbindung herstellen, wird Ihr Amazon DocumentDB-Cluster Ihren Treibern und Clients als Replikatsatz angezeigt. Instances, die Ihrem Amazon DocumentDB-Cluster hinzugefügt und daraus entfernt wurden, werden automatisch in der Konfiguration des Replikatsatzes wiedergegeben.

Jeder Amazon DocumentDB-Cluster besteht aus einem einzelnen Replikatsatz mit dem Standardnamen. `rs0` Der Name des Replikatsatzes kann nicht geändert werden.

Die Verbindung mit dem Cluster-Endpunkt im Replikatsatzmodus ist die empfohlene Methode für den allgemeinen Gebrauch.

> **Note**
>
> Alle Instances in einem Amazon DocumentDB-Cluster überwachen denselben TCP-Port auf Verbindungen.

TLS-Unterstützung

Weitere Informationen zur Verbindung mit Amazon DocumentDB mithilfe von Transport Layer Security (TLS) finden Sie unterVerschlüsseln von Daten während der Übertragung.

Amazon DocumentDB-Speicher

Amazon DocumentDB-Daten werden auf einem Cluster-Volume gespeichert. Dabei handelt es sich um ein einzelnes, virtuelles Volume, das Solid-State-Laufwerke (SSDs) verwendet. Ein Cluster-Volume besteht aus sechs Kopien Ihrer Daten, die automatisch über mehrere Availability Zones gleichzeitig repliziert werden. AWS-Region Diese Replikation trägt dazu bei, dass Ihre Daten sehr langlebig sind und weniger Datenverlust möglich ist. Sie trägt außerdem dazu bei, dass Ihr Cluster während eines Failovers besser verfügbar ist, da Kopien Ihrer Daten bereits in anderen Availability Zones vorhanden sind. Diese Kopien können weiterhin Datenanfragen an die Instances in Ihrem Amazon DocumentDB-Cluster weiterleiten.

Informationen zur Abrechnung des -Datenspeichers

Amazon DocumentDB erhöht automatisch die Größe eines Cluster-Volumes, wenn die Datenmenge zunimmt. Ein Amazon DocumentDB-Cluster-Volume kann auf eine maximale Größe von 256 TiB für Engine-Version 8.0 und höher anwachsen (128 TiB für frühere Engine-Versionen). Ihnen wird jedoch nur der Speicherplatz in Rechnung gestellt, den Sie in einem Amazon DocumentDB-Cluster-Volume verwenden. Ab Amazon DocumentDB 4.0 verringert sich der insgesamt zugewiesene Speicherplatz um einen vergleichbaren Betrag, wenn Daten entfernt werden, z. B. indem eine Sammlung oder ein Index gelöscht wird. Somit können Sie die Speicherkosten senken, indem Sie Sammlungen, Indizes und Datenbanken löschen, die Sie nicht mehr benötigen. In Amazon DocumentDB Version 3.6 kann das Cluster-Volume Speicherplatz wiederverwenden, der beim Entfernen von Daten frei wird, aber das Volume selbst nimmt nie an Größe ab. Daher werden Sie in Version 3.6 möglicherweise keine Änderung des Speichers feststellen, wenn Sie eine Sammlung oder einen Index löschen, obwohl der freigewordene Speicherplatz wiederverwendet wird.

> **Note**
>
> Bei Amazon DocumentDB 3.6 basieren die Speicherkosten auf dem Speicherwert „High Water Mark“ (dem Höchstbetrag, der dem Amazon DocumentDB-Cluster zu einem beliebigen Zeitpunkt zugewiesen wurde). Sie können die Kosten kontrollieren, indem Sie ETL-Praktiken vermeiden, bei denen große Mengen an temporären Informationen erzeugt werden oder bei denen große Mengen neuer Daten geladen werden, bevor nicht mehr benötigte ältere Daten entfernt werden. Wenn das Entfernen von Daten aus einem Amazon DocumentDB-Cluster zu einer beträchtlichen Menge an zugewiesenem, aber ungenutztem Speicherplatz führt, erfordert das Zurücksetzen der Hochwassermarke einen logischen Datendump und die Wiederherstellung auf einem neuen Cluster mithilfe eines Tools wie oder. `mongodump` `mongorestore` Das Erstellen und Wiederherstellen eines Snapshots führt nicht zur

Reduzierung des zugeteilten Speichers, da das physische Layout des zugrunde liegenden Speichers im wiederhergestellten Snapshot unverändert bleibt.

> **Note**
>
> Bei der Nutzung von Hilfsprogrammen wie `mongodump` diesen `mongorestore` fallen I/O Gebühren an, die von der Größe der Daten abhängen, die auf das Speichervolume gelesen und geschrieben werden.

Informationen zur Speicherung und I/O Preisgestaltung von Amazon DocumentDB finden Sie in den häufig gestellten Fragen zu Preisen und Preisen von Amazon DocumentDB (mit MongoDB-Kompatibilität). https://aws.amazon.com/documentdb/faqs/#Pricing

Amazon DocumentDB-Replikation

In einem Amazon DocumentDB-Cluster stellt jede Replikat-Instance einen unabhängigen Endpunkt zur Verfügung. Diese Replikat-Endpunkte bieten Lesezugriff auf die Daten im Cluster-Volume. Mit ihnen können Sie die Leselast für Ihre Daten über mehrere replizierte Instances hinweg skalieren. Sie tragen auch dazu bei, die Leistung beim Lesen von Daten zu verbessern und die Verfügbarkeit der Daten in Ihrem Amazon DocumentDB-Cluster zu erhöhen. Amazon DocumentDB-Repliken sind ebenfalls Failover-Ziele und werden schnell hochgestuft, wenn die primäre Instance für Ihren Amazon DocumentDB-Cluster ausfällt.

Zuverlässigkeit von Amazon DocumentDB

Amazon DocumentDB wurde so konzipiert, dass es zuverlässig, langlebig und fehlertolerant ist. (Um die Verfügbarkeit zu verbessern, sollten Sie Ihren Amazon DocumentDB-Cluster so konfigurieren, dass er über mehrere Replikat-Instances in verschiedenen Availability Zones verfügt.) Amazon DocumentDB enthält mehrere automatische Funktionen, die es zu einer zuverlässigen Datenbanklösung machen.

Automatische Speicherplatzreparatur

Amazon DocumentDB verwaltet mehrere Kopien Ihrer Daten in drei Availability Zones, wodurch die Wahrscheinlichkeit, dass Daten aufgrund eines Speicherausfalls verloren gehen, erheblich reduziert wird. Amazon DocumentDB erkennt automatisch Fehler im Cluster-Volume. Wenn ein Segment eines Cluster-Volumes ausfällt, repariert Amazon DocumentDB das Segment sofort. Es

verwendet die Daten der anderen Volumes, aus denen sich das Cluster-Volumen zusammensetzt, um sicherzustellen, dass die Daten im reparierten Segment aktuell sind. Dadurch vermeidet Amazon DocumentDB Datenverluste und reduziert die Notwendigkeit einer Point-in-Time-Wiederherstellung zur Wiederherstellung nach einem Instance-Ausfall.

Überlebensfähiges Cache-Warming

Amazon DocumentDB verwaltet seinen Seitencache in einem von der Datenbank getrennten Prozess, sodass der Seitencache unabhängig von der Datenbank überleben kann. Im unwahrscheinlichen Fall eines Datenbankausfalls, bleibt der Seiten-Cache im Arbeitsspeicher. Auf diese Weise wird sichergestellt, dass der Pufferpool beim Neustart der Datenbank mit dem aktuellen Zustand vorbereitet wird.

Wiederherstellung nach einem Ausfall

Amazon DocumentDB ist so konzipiert, dass es sich nach einem Absturz fast augenblicklich erholt und Ihre Anwendungsdaten weiterhin bereitstellt. Amazon DocumentDB führt die Wiederherstellung nach einem Absturz asynchron auf parallelen Threads durch, sodass Ihre Datenbank fast unmittelbar nach einem Absturz geöffnet und verfügbar ist.

Verwaltung der Ressourcen

Amazon DocumentDB schützt Ressourcen für kritische Prozesse wie Zustandsprüfungen. Wenn eine Instance einer hohen Speicherbelastung ausgesetzt ist, drosselt Amazon DocumentDB Anfragen. Einige Operationen werden möglicherweise in die Warteschlange gestellt, bis der Speicherdruck nachlässt. Wenn die Speicherauslastung anhält, kann es zu einem Timeout bei Vorgängen in der Warteschlange kommen.

Überwachen Sie anhand der folgenden CloudWatch Messwerte, ob zu wenig Speicherplatz zur Verfügung steht:`LowMemThrottleQueueDepth`,,. `LowMemThrottleMaxQueueDepth` `LowMemNumOperationsThrottled` `LowMemNumOperationsTimedOut` Weitere Informationen finden Sie unter Überwachung von Amazon DocumentDB mit CloudWatch.

Wenn Sie eine anhaltende Speicherauslastung feststellen, skalieren Sie Ihre Instance, um zusätzlichen Speicher für Ihre Arbeitslast bereitzustellen.

Präferenzoptionen lesen

Amazon DocumentDB verwendet einen Cloud-nativen Shared Storage-Service, der Daten sechsmal über drei Availability Zones repliziert, um ein hohes Maß an Haltbarkeit zu gewährleisten. Amazon

DocumentDB ist nicht darauf angewiesen, Daten auf mehrere Instances zu replizieren, um eine lange Lebensdauer zu gewährleisten. Die Daten Ihres Clusters sind beständig, unabhängig davon, ob sie eine einzelne Instance oder 15 Instances enthalten.

Themen

- Haltbarkeit schreiben
- Isolation lesen
- Lesekonsistenz
- Hohe Verfügbarkeit
- Lesevorgänge skalieren

Haltbarkeit schreiben

Amazon DocumentDB verwendet ein einzigartiges, verteiltes, fehlertolerantes, selbstheilendes Speichersystem. Dieses System repliziert sechs Kopien (V=6) Ihrer Daten in drei AWS Availability Zones, um eine hohe Verfügbarkeit und Beständigkeit zu gewährleisten. Beim Schreiben von Daten stellt Amazon DocumentDB sicher, dass alle Schreibvorgänge dauerhaft auf den meisten Knoten aufgezeichnet werden, bevor der Client den Schreibvorgang bestätigt. Wenn Sie einen MongoDB-Replikatsatz mit drei Knoten verwenden, `{w:3, j:true}` würde die Verwendung eines Write-Concern of im Vergleich zu Amazon DocumentDB die bestmögliche Konfiguration ergeben.

Schreibvorgänge in einen Amazon DocumentDB-Cluster müssen von der Writer-Instance des Clusters verarbeitet werden. Der Versuch, in einen Reader zu schreiben, führt zu einem Fehler. Ein bestätigter Schreibvorgang von einer Amazon DocumentDB-Primärinstanz ist dauerhaft und kann nicht rückgängig gemacht werden. Amazon DocumentDB ist standardmäßig sehr robust und unterstützt keine Option für nicht dauerhafte Schreibvorgänge. Sie können die Zuverlässigkeitsstufe (d. h. die Option Write Concern) nicht ändern. Amazon DocumentDB ignoriert w=anything und ist effektiv w: 3 und j: true. Sie können es nicht reduzieren.

Da Speicher und Rechenleistung in der Amazon DocumentDB-Architektur getrennt sind, ist ein Cluster mit einer einzigen Instance äußerst robust. Die Zuverlässigkeit wird auf der Speicherschicht geregelt. Daher erreichen ein Amazon DocumentDB-Cluster mit einer einzelnen Instance und ein Cluster mit drei Instances das gleiche Maß an Haltbarkeit. Sie können Ihren Cluster für Ihren speziellen Anwendungsfall konfigurieren und gleichzeitig für eine hohe Datenbeständigkeit sorgen.

Schreibvorgänge in einen Amazon DocumentDB-Cluster sind innerhalb eines einzelnen Dokuments atomar.

Amazon DocumentDB unterstützt `wtimeout` diese Option nicht und gibt keinen Fehler zurück, wenn ein Wert angegeben wird. Schreibvorgänge in die primäre Amazon DocumentDB-Instance werden garantiert nicht auf unbestimmte Zeit blockiert.

Isolation lesen

Lesevorgänge aus einer Amazon DocumentDB-Instance geben nur Daten zurück, die vor Beginn der Abfrage dauerhaft sind. Lesezugriffe geben niemals Daten zurück, die nach Beginn der Ausführung der Abfrage geändert wurden. Auch "Dirty-Reads" sind unter keinen Umständen möglich.

Lesekonsistenz

Aus einem Amazon DocumentDB-Cluster gelesene Daten sind dauerhaft und werden nicht zurückgesetzt. Sie können die Lesekonsistenz für Amazon DocumentDB-Lesevorgänge ändern, indem Sie die Lesepräferenz für die Anfrage oder Verbindung angeben. Amazon DocumentDB unterstützt keine Option zum Lesen, die nicht dauerhaft ist.

Lesevorgänge aus der primären Instance eines Amazon DocumentDB-Clusters sind unter normalen Betriebsbedingungen sehr konsistent und weisen eine Lese-nach-Schreib-Konsistenz auf. Tritt zwischen dem Schreiben und dem nachfolgenden Lesen ein Failover-Ereignis auf, kann das System kurzzeitig einen nicht "Strongly Consistent"-Wert zurückgeben. Alle Lesezugriffe auf einer gelesenen Replik sind "Eventually Consistent" und geben die Daten in der gleichen Reihenfolge zurück, oft mit weniger als 100 ms Replikationsverzögerung.

Amazon DocumentDB — Leseeinstellungen

Amazon DocumentDB unterstützt das Festlegen einer Lesepräferenzoption nur, wenn Daten vom Cluster-Endpunkt im Replikatsatzmodus gelesen werden. Das Festlegen einer Lesepräferenzoption wirkt sich darauf aus, wie Ihr MongoDB-Client oder -Treiber Leseanfragen an Instances in Ihrem Amazon DocumentDB-Cluster weiterleitet. Sie können Leseeinstellungen für eine bestimmte Abfrage oder als allgemeine Option in Ihrem MongoDB-Treiber festlegen. (Lesen Sie in der Dokumentation Ihres Clients oder Treibers nach, wie Sie eine Leseeinstellung festlegen können.)

Wenn Ihr Client oder Treiber im Replica-Set-Modus keine Verbindung zu einem Amazon DocumentDB-Cluster-Endpunkt herstellt, ist das Ergebnis der Angabe einer Lesepräferenz undefiniert.

Amazon DocumentDB unterstützt das Festlegen von Tag-Sets als Lesepräferenz nicht.

Unterstützte Optionen für Leseeinstellungen

- **`primary`**— Durch die Angabe einer `primary` Lesepräferenz wird sichergestellt, dass alle Lesevorgänge an die primäre Instanz des Clusters weitergeleitet werden. Wenn die primäre Instance nicht verfügbar ist, schlägt der Lesevorgang fehl. Die Leseeinstellung `primary` ergibt Lese-After-Write-Konsistenz und eignet sich für Anwendungsfälle, die die Lese-nach-Schreib-Konsistenz über Hochverfügbarkeit und Leseskalierung priorisieren.

  Im folgenden Beispiel wird die Leseeinstellung `primary` angegeben:

  ```
  db.example.find().readPref('primary')
  ```

- **`primaryPreferred`**— Wenn `primaryPreferred` Sie eine Lesepräferenz angeben, werden Lesevorgänge im Normalbetrieb an die primäre Instance weitergeleitet. Wenn es ein primäres Failover gibt, leitet der Client Anfragen an ein Replikat weiter. Die Leseeinstellung `primaryPreferred` sorgt für Lese-nach-Schreib-Konsistenz im Normalbetrieb und Eventually Consistent-Lesezugriffe während eines Failoverereignisses. Die Leseeinstellung `primaryPreferred` ist für Anwendungsfälle geeignet, die die Lese-nach-Schreib-Konsistenz über Hochverfügbarkeit und Leseskalierung priorisieren, jedoch trotzdem eine Hochverfügbarkeit erfordern.

  Im folgenden Beispiel wird die Leseeinstellung `primaryPreferred` angegeben:

  ```
  db.example.find().readPref('primaryPreferred')
  ```

- **`secondary`**— Durch die Angabe einer `secondary` Lesepräferenz wird sichergestellt, dass Lesevorgänge nur an ein Replikat weitergeleitet werden, niemals an die primäre Instanz. Wenn es in einem Cluster keine Replikat-Instances gibt, schlägt die Leseanforderung fehl. Die Leseeinstellung `secondary` ergibt Eventually Consistent-Lesezugriffe und ist für Anwendungsfälle geeignet, die den Schreibdurchsatz der primären Instance über die Hochverfügbarkeit und die Lesen-nach-Schreiben-Konsistenz priorisiert.

  Im folgenden Beispiel wird die Leseeinstellung `secondary` angegeben:

  ```
  db.example.find().readPref('secondary')
  ```

- **`secondaryPreferred`**— Durch die Angabe einer `secondaryPreferred` Lesepräferenz wird sichergestellt, dass Lesevorgänge an ein Lesereplikat weitergeleitet werden, wenn ein oder mehrere Replikate aktiv sind. Wenn es in einem Cluster keine aktiven Replikat-Instances gibt, wird die Leseanforderung an die primäre Instance weitergeleitet. Die Leseeinstellung `secondaryPreferred` ergibt Eventually Consistent-Lesezugriffe, wenn der Lesezugriff durch ein Read Replica bedient wird. Sie liefert eine Lese-nach-Schreib-Konsistenz, wenn das Lesen von der primären Instance bedient wird (mit Ausnahme von Failover-Ereignissen). Die Leseeinstellung `secondaryPreferred` ist für Anwendungsfälle geeignet, die Leseskalierung und Hochverfügbarkeit über Lese-nach-Schreib-Konsistenz priorisieren.

  Im folgenden Beispiel wird die Leseeinstellung `secondaryPreferred` angegeben:

  ```
  db.example.find().readPref('secondaryPreferred')
  ```

- **`nearest`**— Die Angabe einer `nearest` Lesepräferenz leitet Lesevorgänge ausschließlich auf der Grundlage der gemessenen Latenz zwischen dem Client und allen Instances im Amazon DocumentDB-Cluster weiter. Die Leseeinstellung `nearest` ergibt Eventually Consistent-Lesezugriffe, wenn der Lesezugriff durch ein Read Replica bedient wird. Sie liefert eine Lese-nach-Schreib-Konsistenz, wenn das Lesen von der primären Instance bedient wird (mit Ausnahme von Failover-Ereignissen). Die Leseeinstellung `nearest` ist für Anwendungsfälle geeignet, die das Erreichen einer möglichst geringen Leselatenz und die Hochverfügbarkeit über die Lese-zu-Schreib-Konsistenz und Leseskalierung priorisieren.

  Im folgenden Beispiel wird die Leseeinstellung `nearest` angegeben:

  ```
  db.example.find().readPref('nearest')
  ```

Hohe Verfügbarkeit

Amazon DocumentDB unterstützt hochverfügbare Cluster-Konfigurationen, indem Repliken als Failover-Ziele für die primäre Instance verwendet werden. Wenn die primäre Instance ausfällt, wird eine Amazon DocumentDB-Replica als neue primäre Instance hochgestuft. Dabei gibt es eine kurze Unterbrechung, in der Lese- und Schreibanforderungen an die primäre Instance mit einer Ausnahme fehlschlagen.

Wenn Ihr Amazon DocumentDB-Cluster keine Replikate enthält, wird die primäre Instance bei einem Ausfall neu erstellt. Da das Heraufstufen eines Amazon DocumentDB-Replikats viel schneller ist als das Neuerstellen der primären Instance, erstellen Sie eine oder mehrere Amazon DocumentDB-Repliken als Failover-Ziele.

Replikate, die als Failover-Ziele verwendet werden sollen, sollten dieselbe DB-Instance-Klasse haben wie die primäre Instance. Sie sollten von der primären Instance in verschiedenen Availability Zones bereitgestellt werden. Sie können steuern, welche Replikate als Failover-Ziele bevorzugt werden. Bewährte Methoden zur Konfiguration von Amazon DocumentDB für hohe Verfügbarkeit finden Sie unter. Grundlegendes zur Amazon DocumentDB-Cluster-Fehlertoleranz

Lesevorgänge skalieren

Amazon DocumentDB-Repliken eignen sich ideal für die Skalierung von Lesevorgängen. Sie sind in Ihrem Cluster-Volume vollständig auf Lesevorgänge ausgerichtet, d. h. Replikate verarbeiten keine Schreibvorgänge. Die Datenreplikation geschieht innerhalb des Cluster-Volumes und nicht zwischen Instances. Deshalb sind die Replikat-Ressourcen auf die Verarbeitung von Abfragen ausgelegt und nicht auf das Schreiben und Replizieren von Daten.

Wenn Ihre Anwendung mehr Lesekapazität benötigt, können Sie Ihrem Cluster schnell (in der Regel in weniger als zehn Minuten) eine Replik hinzufügen. Wenn Ihr Lesekapazitätsbedarf sinkt, können Sie nicht benötigte Replikate entfernen. Bei Amazon DocumentDB-Repliken zahlen Sie nur für die Lesekapazität, die Sie benötigen.

Amazon DocumentDB unterstützt die clientseitige Leseskalierung durch die Verwendung von Read Preference-Optionen. Weitere Informationen finden Sie unter Amazon DocumentDB — Leseeinstellungen.

TTL löscht

Löschungen aus einem TTL-Indexbereich über einen Hintergrundprozess erfolgen nach dem Best-Effort-Prinzip und können nicht für einen bestimmten Zeitrahmen garantiert werden. Faktoren wie Instance-Größe, Instance-Ressourcenauslastung, Dokumentgröße und Gesamtdurchsatz können sich auf den Zeitpunkt einer TTL-Löschung auswirken.

Wenn der TTL-Monitor Ihre Dokumente löscht, entstehen bei jeder Löschung E/A-Kosten, was den Rechnungsbetrag erhöht. Wenn der Durchsatz und die TTL-Löschraten steigen, sollten Sie aufgrund der erhöhten I/O-Nutzung mit einer Erhöhung Ihrer Rechnung rechnen.

Wenn Sie einen TTL-Index für eine vorhandene Sammlung erstellen, müssen Sie alle abgelaufenen Dokumente löschen, bevor Sie den Index erstellen. Die aktuelle TTL-Implementierung ist für das Löschen eines kleinen Bruchteils der Dokumente in der Sammlung optimiert. Dies ist typisch, wenn TTL für die Sammlung von Anfang an aktiviert war. Dies kann zu höheren IOPS als nötig führen, wenn eine große Anzahl von Dokumenten auf einmal gelöscht werden muss.

Wenn Sie keinen TTL-Index zum Löschen von Dokumenten erstellen möchten, können Sie Dokumente stattdessen nach Zeit in Sammlungen unterteilen und diese Sammlungen einfach löschen, wenn die Dokumente nicht mehr benötigt werden. Beispiel: Sie können eine Sammlung pro Woche erstellen und diese löschen, ohne dass IO-Kosten anfallen. Dies kann erheblich kostengünstiger sein als die Verwendung eines TTL-Index.

Abrechnungsfähige Ressourcen

Identifizierung fakturierbarer Amazon DocumentDB-Ressourcen

Als vollständig verwalteter Datenbankservice erhebt Amazon DocumentDB Gebühren für Instances, Speicher I/Os, Backups und Datenübertragung. Weitere Informationen finden Sie unter Preise für Amazon DocumentDB (mit MongoDB-Kompatibilität).

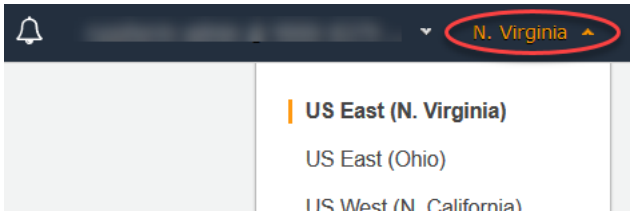
Um abrechnungsfähige Ressourcen in Ihrem Konto zu ermitteln und die Ressourcen möglicherweise zu löschen, können Sie das oder verwenden. AWS-Managementkonsole AWS CLI

Verwendung der AWS-Managementkonsole

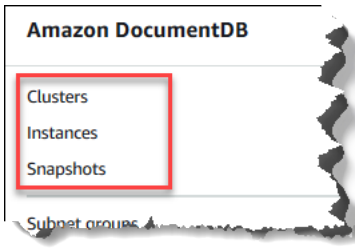
Mithilfe von können Sie die AWS-Managementkonsole Amazon DocumentDB-Cluster, -Instances und -Snapshots ermitteln, die Sie für eine bestimmte Zeit bereitgestellt haben. AWS-Region

So ermitteln Sie Cluster, Instances und Snapshots:

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Um kostenpflichtige Ressourcen in einer anderen Region als Ihrer Standardregion zu finden, wählen Sie in der oberen rechten Ecke des Bildschirms die Region aus, nach der AWS-Region Sie suchen möchten.



3. Wählen Sie im Navigationsbereich die Art der kostenpflichtigen Ressource aus: Clusters (Cluster), Instances oder Snapshots.



4. Im rechten Bereich werden alle bereitgestellten Cluster, Instances oder Snapshots für die Region aufgelistet. Für Cluster, Instances und Snapshots werden Gebühren berechnet.

Verwendung der AWS CLI

Mithilfe von können Sie die AWS CLI Amazon DocumentDB-Cluster, Instances und Snapshots ermitteln, die Sie für eine bestimmte Zeit bereitgestellt haben. AWS-Region

So ermitteln Sie Cluster und Instances:

Der folgende Code listet Ihre gesamten Cluster und Instances für die angegebene Region auf. Wenn Sie nach Clustern und Instances in Ihrer Standardregion suchen möchten, können Sie den Parameter `--region` weglassen.

Example

Für Linux, macOS oder Unix:

```
aws docdb describe-db-clusters \
    --region us-east-1 \
    --query 'DBClusters[?Engine==`docdb`]' | \
       grep -e "DBClusterIdentifier" -e "DBInstanceIdentifier"
```

Für Windows:

```
aws docdb describe-db-clusters ^
    --region us-east-1 ^
    --query 'DBClusters[?Engine==`docdb`]' | ^
       grep -e "DBClusterIdentifier" -e "DBInstanceIdentifier"
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
"DBClusterIdentifier": "docdb-2019-01-09-23-55-38",
        "DBInstanceIdentifier": "docdb-2019-01-09-23-55-38",
        "DBInstanceIdentifier": "docdb-2019-01-09-23-55-382",
"DBClusterIdentifier": "sample-cluster",
"DBClusterIdentifier": "sample-cluster2",
```

So ermitteln Sie Snapshots:

Der folgende Code listet Ihre gesamten Snapshots für die angegebene Region auf. Wenn Sie in Ihrer Standardregion nach Snapshots suchen möchten, können Sie den Parameter `--region` weglassen.

Für Linux, macOS oder Unix:

```
aws docdb describe-db-cluster-snapshots \
  --region us-east-1 \
  --query 'DBClusterSnapshots[?Engine==`docdb`].
[DBClusterSnapshotIdentifier,SnapshotType]'
```

Für Windows:

```
aws docdb describe-db-cluster-snapshots ^
  --region us-east-1 ^
  --query 'DBClusterSnapshots[?Engine==`docdb`].
[DBClusterSnapshotIdentifier,SnapshotType]'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
[
    [
        "rds:docdb-2019-01-09-23-55-38-2019-02-13-00-06",
        "automated"
    ],
    [
        "test-snap",
        "manual"
    ]
]
```

Sie müssen nur `manual`-Snapshots löschen. `Automated`-Snapshots werden gelöscht, wenn Sie den Cluster löschen.

Löschen unerwünschter fakturierbarer Ressourcen

Um einen Cluster zu löschen, müssen Sie zunächst alle Instances im Cluster löschen.

- Weitere Informationen zum Löschen von Instances finden Sie unter Löschen einer Amazon DocumentDB DocumentDB-Instance.

> **Important**
>
> Auch wenn Sie die Instances in einem Cluster löschen, wird Ihnen die mit diesem Cluster verbundene Speicher- und Sicherungsnutzung in Rechnung gestellt. Um alle Kosten zu stoppen, müssen Sie auch Ihren Cluster und manuelle Snapshots löschen.

- Weitere Informationen zum Löschen von Clustern finden Sie unter Löschen eines Amazon DocumentDB-Clusters.
- Weitere Informationen zum Löschen manueller Snapshots finden Sie unter Löschen eines Cluster-Snapshots.

Was ist eine Dokumentendatenbank?

Einige Entwickler betrachten ihr Datenmodell nicht als normalisierte Zeilen und Spalten. In der Regel werden Daten auf Anwendungsebene als JSON-Dokument dargestellt, da es für Entwickler intuitiver ist, ihr Datenmodell als Dokument zu betrachten.

Die Popularität von Dokumentdatenbanken ist gestiegen, weil Sie mit ihnen Daten in einer Datenbank erhalten können, indem Sie das gleiche Dokumentmodellformat verwenden, das Sie in Ihrem Anwendungscode verwenden. Dokumentdatenbanken bieten leistungsfähige und intuitive APIs für eine flexible und agile Entwicklung.

Themen

- Anwendungsfälle für Dokumentendatenbanken
- Dokumente verstehen
- Arbeiten mit Dokumenten

Anwendungsfälle für Dokumentendatenbanken

Ihr Anwendungsfall bestimmt, ob Sie eine Dokumentendatenbank oder eine andere Art von Datenbank für die Verwaltung Ihrer Daten benötigen. Dokumentdatenbanken sind nützlich für Workloads, die ein flexibles Schema für eine schnelle, iterative Entwicklung benötigen. Im Folgenden finden Sie einige Beispiele für Anwendungsfälle, bei denen Dokumentendatenbanken erhebliche Vorteile bieten können:

Themen

- Benutzerprofile
- Real-time große Datenmengen
- Verwaltung von Inhalten

Benutzerprofile

Da Dokumentdatenbanken über ein flexibles Schema verfügen, können sie Dokumente speichern, die unterschiedliche Attribute und Datenwerte aufweisen. Dokumentdatenbanken sind eine praktische Lösung für Online-Profile, in denen verschiedene Benutzer unterschiedliche Arten von Informationen bereitstellen. Mit Hilfe einer Dokumentdatenbank können Sie das Profil jedes Benutzers effizient speichern, indem Sie nur die Attribute speichern, die für jeden Benutzer spezifisch sind.

Angenommen, ein Benutzer entscheidet sich dafür, Informationen in seinem Profil hinzuzufügen oder zu entfernen. In diesem Fall könnte ihr Dokument leicht durch eine aktualisierte Version ersetzt werden, die alle kürzlich hinzugefügten Attribute und Daten enthält oder alle neu weggelassenen Attribute und Daten auslässt. Dokumentdatenbanken bewältigen diese Individualität und Fluidität auf einfache Weise.

Real-time große Datenmengen

In der Vergangenheit wurde die Fähigkeit, Informationen aus Betriebsdaten zu extrahieren, dadurch beeinträchtigt, dass operative Datenbanken und analytische Datenbanken in unterschiedlichen Umgebungen verwaltet wurden — den betrieblichen bzw. den jeweiligen. business/reporting Die Fähigkeit, operative Informationen in Echtzeit zu extrahieren, ist in einem hart umkämpften Geschäftsumfeld von entscheidender Bedeutung. Durch die Verwendung von Dokumentendatenbanken kann ein Unternehmen Betriebsdaten aus jeder beliebigen Quelle speichern und verwalten und gleichzeitig die Daten zur Analyse an die BI-Engine seiner Wahl weiterleiten. Es ist nicht erforderlich, dass zwei Umgebungen vorhanden sein müssen.

Verwaltung von Inhalten

Um Inhalte effektiv zu verwalten, müssen Sie in der Lage sein, Inhalte aus einer Vielzahl von Quellen zu sammeln, zu aggregieren und dann an den Client zu liefern. Aufgrund ihres flexiblen Schemas sind Dokumentdatenbanken ideal für die Erfassung und Speicherung jeglicher Art von Daten. Mit ihnen können Sie neue Arten von Inhalten erstellen und integrieren, einschließlich benutzergenerierter Inhalte wie Bilder, Kommentare und Videos.

Dokumente verstehen

Dokumentendatenbanken werden verwendet, um semistrukturierte Daten als Dokument zu speichern, anstatt Daten in mehreren Tabellen zu normalisieren, von denen jede eine eindeutige und feste Struktur hat, wie in einer relationalen Datenbank. Dokumente, die in einer Dokumentdatenbank gespeichert sind, verwenden verschachtelte Schlüssel-Werte-Paare, um die Struktur oder das Schema des Dokuments bereitzustellen. Es können jedoch verschiedene Arten von Dokumenten in derselben Dokumentdatenbank gespeichert werden, wodurch die Anforderung erfüllt wird, ähnliche Daten in unterschiedlichen Formaten zu verarbeiten. Da beispielsweise jedes Dokument selbsterklärend ist, Beispieldokumente in einer Dokumentendatenbank können die im Thema beschriebenen JSON-encoded Dokumente für einen Onlineshop in derselben Dokumentendatenbank gespeichert werden.

Themen

- SQL im Vergleich zu nichtrelationaler Terminologie
- Einfache Dokumente
- Eingebettete Dokumente
- Beispieldokumente in einer Dokumentendatenbank
- Grundlegendes zur Normalisierung in einer Dokumentendatenbank

SQL im Vergleich zu nichtrelationaler Terminologie

Die folgende Tabelle vergleicht die von Dokumentendatenbanken (MongoDB) verwendete Terminologie mit der von SQL-Datenbanken verwendeten Terminologie.

SQL	MongoDB
Tabelle	Sammlung

SQL	MongoDB
Zeile	Dokument
Spalte	Feld
Primärschlüssel	ObjectId
Index	Index
Anzeigen	Anzeigen
Verschachtelte(s) Tabelle oder Objekt	Eingebettetes Dokument
Array	Array

Einfache Dokumente

Alle Dokumente in einer Dokumentendatenbank sind selbstbeschreibend. In dieser Dokumentation JSON-like werden formatierte Dokumente verwendet, Sie können jedoch auch andere Kodierungsmethoden verwenden.

Ein einfaches Dokument hat ein oder mehrere Felder, die alle auf der gleichen Ebene innerhalb des Dokuments liegen. Im folgenden Beispiel sind die Felder `SSN`, `LName`, `FName`, `DOB`, `Street`, `City`, `State-Province`, `PostalCode` und `Country` alle Elemente auf derselben Ebene im Dokument.

```
{
   "SSN": "123-45-6789",
   "LName": "Rivera",
   "FName": "Martha",
   "DOB": "1992-11-16",
   "Street": "125 Main St.",
   "City": "Anytown",
   "State-Province": "WA",
   "PostalCode": "98117",
   "Country": "USA"
}
```

Wenn Informationen in einem einfachen Dokument organisiert sind, wird jedes Feld einzeln verwaltet. Um die Adresse einer Person abzurufen, müssen Sie `Street`, `City`, `State-Province`, `PostalCode` und `Country` als einzelne Datenelemente abrufen.

Eingebettete Dokumente

Ein komplexes Dokument organisiert seine Daten, indem es eingebettete Dokumente innerhalb des Dokuments erstellt. Eingebettete Dokumente helfen bei der Verwaltung von Daten in Gruppierungen und als einzelne Datenelemente, je nachdem, was im jeweiligen Fall effizienter ist. Mit dem vorhergehenden Beispiel könnten Sie ein `Address`-Dokument in das Hauptdokument einbetten. Dadurch ergibt sich die folgende Dokumentstruktur:

```
{
   "SSN": "123-45-6789",
   "LName": "Rivera",
   "FName": "Martha",
   "DOB": "1992-11-16",
   "Address":
   {
       "Street": "125 Main St.",
       "City": "Anytown",
       "State-Province": "WA",
       "PostalCode": "98117",
       "Country": "USA"
   }
}
```

Sie können jetzt auf die Daten im Dokument als einzelne Felder (`"SSN":`), als eingebettetes Dokument (`"Address":`) oder als Mitglied eines eingebetteten Dokuments (`"Address": {"Street":}`) zugreifen.

Beispieldokumente in einer Dokumentendatenbank

Da jedes Dokument in einer Dokumentdatenbank selbstbeschreibend ist, kann die Struktur von Dokumenten innerhalb einer Dokumentdatenbank unterschiedlich sein. Die folgenden beiden Dokumente, eines für ein Buch und eines für eine Zeitschrift, sind strukturell unterschiedlich. Beide können sich jedoch in der gleichen Dokumentdatenbank befinden.

Im Folgenden finden Sie ein Beispiel-Buch-Dokument:

```
{
```

```
    "_id" : "9876543210123",
    "Type": "book",
    "ISBN": "987-6-543-21012-3",
    "Author":
    {
        "LName":"Roe",
        "MI": "T",
        "FName": "Richard"
    },
    "Title": "Understanding Document Databases"
}
```

Im Folgenden finden Sie ein Beispiel für ein Zeitschriften-Dokument mit zwei Artikeln:

```
{
    "_id" : "0123456789012",
    "Publication": "Programming Today",
    "Issue":
    {
        "Volume": "14",
        "Number": "09"
    },
    "Articles" : [
        {
            "Title": "Is a Document Database Your Best Solution?",
            "Author":
            {
                "LName": "Major",
                "FName": "Mary"
            }
        },
        {
            "Title": "Databases for Online Solutions",
            "Author":
            {
                "LName": "Stiles",
                "FName": "John"
            }
        }
    ],
    "Type": "periodical"
}
```

Vergleichen Sie die Struktur dieser beiden Dokumente. Bei einer relationalen Datenbank benötigen Sie entweder getrennte Tabellen "Zeitschriften" und "Bücher" oder eine einzelne Tabelle mit unbenutzten Feldern wie "Publikation", "Ausgabe", "Artikel" und "MI" als `null` Werte. Da Dokumentendatenbanken halbstrukturiert sind und jedes Dokument seine eigene Struktur definiert, können diese beiden Dokumente ohne `null`-Felder gleichzeitig in derselben Dokumentendatenbank vorhanden sein. Dokumentdatenbanken sind gut geeignet, um mit lückenhaften Daten umzugehen.

Die Entwicklung auf Basis einer Dokumentdatenbank ermöglicht eine schnelle, iterative Entwicklung. Denn Sie können die Datenstruktur eines Dokuments dynamisch ändern, ohne das Schema für die gesamte Sammlung ändern zu müssen. Dokumentdatenbanken eignen sich hervorragend für agile Entwicklungen und sich dynamisch verändernde Umgebungen.

Grundlegendes zur Normalisierung in einer Dokumentendatenbank

Dokumentdatenbanken werden nicht normalisiert; in einem Dokument gefundene Daten können in einem anderen Dokument wiederholt werden. Darüber hinaus können einige Datenunterschiede zwischen den Dokumenten bestehen. Betrachten Sie beispielsweise das Szenario, in dem Sie einen Einkauf in einem Online-Shop tätigen und alle Details Ihrer Einkäufe in einem einzigen Dokument gespeichert sind. Das Dokument könnte etwa so aussehen wie das folgende JSON-Dokument:

```
{
    "DateTime": "2018-08-15T12:13:10Z",
    "LName" : "Santos",
    "FName" : "Paul",
    "Cart" : [
        {
            "ItemId" : "9876543210123",
            "Description" : "Understanding Document Databases",
            "Price" : "29.95"
        },
        {
            "ItemId" : "0123456789012",
            "Description" : "Programming Today",
            "Issue": {
                "Volume": "14",
                "Number": "09"
            },
            "Price" : "8.95"
        },
        {
            "ItemId": "234567890-K",
```

```
            "Description": "Gel Pen (black)",
            "Price": "2.49"
        }
    ],
    "PaymentMethod" :
    {
        "Issuer" : "MasterCard",
        "Number" : "1234-5678-9012-3456"
    },
    "ShopperId" : "1234567890"
}
```

Alle diese Informationen werden als Dokument in einer Transaktionssammlung gespeichert. Später merken Sie, dass Sie vergessen haben, einen Artikel zu kaufen. Daher melden Sie sich erneut im gleichen Shop an und tätigen einen weiteren Kauf, der auch als ein weiteres Dokument in der Transaktionssammlung gespeichert wird.

```
{
    "DateTime": "2018-08-15T14:49:00Z",
    "LName" : "Santos",
    "FName" : "Paul",
    "Cart" : [
        {
            "ItemId" : "2109876543210",
            "Description" : "Document Databases for Fun and Profit",
            "Price" : "45.95"
        }
    ],
    "PaymentMethod" :
    {
        "Issuer" : "Visa",
        "Number" : "0987-6543-2109-8765"
    },
    "ShopperId" : "1234567890"
}
```

Beachten Sie die Redundanz zwischen diesen beiden Dokumenten — Ihrem Namen und Ihrer Kunden-ID (und, falls Sie dieselbe Kreditkarte verwendet haben, Ihren Kreditkarteninformationen). Aber das ist in Ordnung, denn die Speicherung ist kostengünstig, und jedes Dokument zeichnet eine einzelne Transaktion vollständig auf, die mit einer einfachen Schlüssel-/Wertabfrage, die keine Verknüpfungen erfordert, schnell abgerufen werden kann.

Es besteht auch eine offensichtliche Diskrepanz zwischen den beiden Dokumenten — Ihren Kreditkarteninformationen. Dies ist nur eine scheinbare Diskrepanz, da es wahrscheinlich ist, dass Sie für jeden Kauf eine andere Kreditkarte verwendet haben. Jedes Dokument entspricht genau der Transaktion, die es dokumentiert.

Arbeiten mit Dokumenten

Als Dokumentendatenbank macht es Amazon DocumentDB einfach, JSON-Daten zu speichern, abzufragen und zu indizieren. In Amazon DocumentDB entspricht eine Sammlung einer Tabelle in einer relationalen Datenbank, außer dass kein einziges Schema für alle Dokumente durchgesetzt wird. Mit Sammlungen können Sie ähnliche Dokumente gruppieren und gleichzeitig alle in derselben Datenbank halten, ohne dass sie in ihrer Struktur identisch sein müssen.

Unter Verwendung der Beispieldokumente aus früheren Abschnitten ist es wahrscheinlich, dass Sie Sammlungen für `reading_material` und `office_supplies` haben werden. Es liegt in der Verantwortung Ihrer Software, die Zugehörigkeit eines Dokuments zu einer bestimmten Sammlung durchzusetzen.

Die folgenden Beispiele zeigen anhand der MongoDB-API, wie Sie Dokumente hinzufügen, abfragen, aktualisieren und löschen können.

Themen

- Dokumente hinzufügen
- Dokumente werden abgefragt
- Dokumente werden aktualisiert
- Dokumente löschen

Dokumente hinzufügen

In Amazon DocumentDB wird eine Datenbank erstellt, wenn Sie ein Dokument zum ersten Mal zu einer Sammlung hinzufügen. In diesem Beispiel erstellen Sie eine Sammlung mit dem Namen `example` in der Datenbank `test`, die die Standarddatenbank ist, wenn Sie eine Verbindung mit einem Cluster herstellen. Da die Sammlung implizit erstellt wird, wenn das erste Dokument eingefügt wird, gibt es bei der Überprüfung des Sammlungsnamens keine Fehler. Das heißt, ein Tippfehler im Sammlungsnamen, wie z. B. `eexample` statt `example` führt dazu, dass das Dokument erstellt und der Sammlung `eexample` und nicht der beabsichtigten Sammlung hinzugefügt wird. Die Fehlerüberprüfung muss von Ihrer Anwendung durchgeführt werden.

Die folgenden Beispiele verwenden die MongoDB-API zum Hinzufügen der Dokumente.

Themen

- Ein einzelnes Dokument hinzufügen
- Hinzufügen mehrerer Dokumente

Ein einzelnes Dokument hinzufügen

Um ein einzelnes Dokument zu einer Sammlung hinzuzufügen, verwenden Sie die Operation `insertOne( {} )` mit dem Dokument, das Sie der Sammlung hinzufügen möchten.

```
db.example.insertOne(
    {
        "Item": "Ruler",
        "Colors": ["Red","Green","Blue","Clear","Yellow"],
        "Inventory": {
            "OnHand": 47,
            "MinOnHand": 40
        },
        "UnitPrice": 0.89
    }
)
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    "acknowledged" : true,
    "insertedId" : ObjectId("5bedafbcf65ff161707de24f")
}
```

Hinzufügen mehrerer Dokumente

Um mehrere Dokumente zu einer Sammlung hinzuzufügen, verwenden Sie die Operation `insertMany( [{},...,{}] )` mit einer Liste der Dokumente, die Sie der Sammlung hinzufügen möchten. Obwohl die Dokumente in dieser Liste unterschiedliche Schemata haben, können sie alle zur gleichen Sammlung hinzugefügt werden.

```
db.example.insertMany(
    [
```

```
        {
            "Item": "Pen",
            "Colors": ["Red","Green","Blue","Black"],
            "Inventory": {
                "OnHand": 244,
                "MinOnHand": 72
            }
        },
        {
            "Item": "Poster Paint",
            "Colors": ["Red","Green","Blue","Black","White"],
            "Inventory": {
                "OnHand": 47,
                "MinOnHand": 50
            }
        },
        {
            "Item": "Spray Paint",
            "Colors": ["Black","Red","Green","Blue"],
            "Inventory": {
                "OnHand": 47,
                "MinOnHand": 50,
                "OrderQnty": 36
            }
        }
    ]
)
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    "acknowledged" : true,
    "insertedIds" : [
            ObjectId("5bedb07941ca8d9198f5934c"),
            ObjectId("5bedb07941ca8d9198f5934d"),
            ObjectId("5bedb07941ca8d9198f5934e")
    ]
}
```

Dokumente werden abgefragt

Manchmal müssen Sie möglicherweise den Bestand Ihres Online-Shops nachschlagen, damit Kunden das Angebot sehen und kaufen können. Die Abfrage einer Sammlung ist relativ einfach,

unabhängig davon, ob Sie alle Dokumente in der Sammlung haben möchten oder nur die Dokumente, die ein bestimmtes Kriterium erfüllen.

Verwenden Sie die Operation `find()`, um Dokumente abzufragen. Der Befehl `find()` hat einen einzigen Dokumentenparameter, der die Kriterien für die Auswahl der zurückzugebenden Dokumente definiert. Die Ausgabe von `find()` ist ein Dokument, das als einzelne Textzeile ohne Zeilenumbrüche formatiert ist. Um das Ausgabedokument für eine bessere Lesbarkeit zu formatieren, verwenden Sie `find().pretty()`. Alle Beispiele in diesem Thema verwenden `.pretty()` zum Formatieren der Ausgabe.

Verwenden Sie die vier Dokumente, die Sie in den beiden vorherigen Übungen in die `example` Sammlung eingefügt haben — `insertOne()` und`insertMany()`.

Themen

- Alle Dokumente in einer Sammlung abrufen
- Dokumente werden abgerufen, die einem Feldwert entsprechen
- Dokumente werden abgerufen, die einem eingebetteten Dokument entsprechen
- Dokumente werden abgerufen, die einem Feldwert in einem eingebetteten Dokument entsprechen
- Dokumente werden abgerufen, die einem Array entsprechen
- Dokumente werden abgerufen, die einem Wert in einem Array entsprechen
- Dokumente mithilfe von Operatoren abrufen

Alle Dokumente in einer Sammlung abrufen

Um alle Dokumente in Ihrer Sammlung abzurufen, verwenden Sie die Operation `find()` mit einem leeren Abfragedokument.

Die folgende Abfrage gibt alle Dokumente der Sammlung `example` zurück.

```
db.example.find( {} ).pretty()
```

Dokumente werden abgerufen, die einem Feldwert entsprechen

Um alle Dokumente abzurufen, die mit einem Feld und einem Wert übereinstimmen, verwenden Sie die Operation `find()` mit einem Abfragedokument, das die entsprechenden Felder und Werte identifiziert.

Bei Verwendung der vorangegangenen Dokumente gibt diese Abfrage alle Dokumente zurück, bei denen das Feld "Item" (Element) "Pen" (Stift) entspricht.

```
db.example.find( { "Item": "Pen" } ).pretty()
```

Dokumente werden abgerufen, die einem eingebetteten Dokument entsprechen

Um alle Dokumente zu suchen, die mit einem eingebetteten Dokument übereinstimmen, verwenden Sie die Operation `find()` mit einem Abfragedokument, in dem der Name des eingebetteten Dokuments sowie alle Felder und Werte für dieses eingebettete Dokument angegeben werden.

Beim Vergleichen mit einem eingebetteten Dokument muss das eingebettete Dokument denselben Namen haben wie in der Abfrage. Zudem müssen die Felder und Werte im eingebetteten Dokument mit der Abfrage übereinstimmen.

Die folgende Abfrage gibt nur das Dokument "Poster Paint" zurück. Dies liegt daran, dass "Pen" über verschiedene Werte für "`OnHand`" und "`MinOnHand`" verfügt und "Spray Paint" ein weiteres Feld (`OrderQnty`) als das Abfragedokument besitzt.

```
db.example.find({"Inventory": {
    "OnHand": 47,
    "MinOnHand": 50 } } ).pretty()
```

Dokumente werden abgerufen, die einem Feldwert in einem eingebetteten Dokument entsprechen

Um alle Dokumente zu suchen, die mit einem eingebetteten Dokument übereinstimmen, verwenden Sie die Operation `find()` mit einem Abfragedokument, in dem der Name des eingebetteten Dokuments sowie alle Felder und Werte für dieses eingebettete Dokument angegeben werden.

Aufgrund der vorangegangenen Dokumente verwendet die folgende Abfrage "Punktnotation", um das eingebettete Dokument und die Felder von Interesse anzugeben. Jedes Dokument, das damit übereinstimmt, wird zurückgegeben, unabhängig davon, welche anderen Felder im eingebetteten Dokument vorhanden sind. Die Abfrage gibt "Poster Paint" und "Spray Paint" zurück, weil sie beide den angegebenen Feldern und Werten entsprechen.

```
db.example.find({"Inventory.OnHand": 47, "Inventory.MinOnHand": 50 }).pretty()
```

Dokumente werden abgerufen, die einem Array entsprechen

Um alle Dokumente zu finden, die einem Array entsprechen, verwenden Sie die Operation `find()` mit dem Namen des Arrays, an dem Sie interessiert sind, und allen Werten in diesem Array. Die Abfrage gibt alle Dokumente zurück, in denen sich ein Array mit diesem Namen befindet und in denen die Array-Werte identisch sind und die gleiche Reihenfolge wie in der Abfrage aufweisen.

Die folgende Abfrage gibt nur "Pen" zurück, da "Poster Paint" über eine zusätzlichen Farbe (White) verfügt und die Farben in "Spray Paint" in einer anderen Reihenfolge vorliegen.

```
db.example.find( { "Colors": ["Red","Green","Blue","Black"] } ).pretty()
```

Dokumente werden abgerufen, die einem Wert in einem Array entsprechen

Um alle Dokumente mit einem bestimmten Array-Wert zu finden, verwenden Sie die Operation `find()` mit dem Namen und Wert des Arrays, an dem Sie interessiert sind.

```
db.example.find( { "Colors": "Red" } ).pretty()
```

Bei der vorherigen Operation werden alle drei Dokumente zurückgegeben, da jedes davon ein Array mit dem Namen `Colors` und den Wert "`Red`" irgendwo im Array besitzt. Wenn Sie den Wert "`White`" angeben, gibt die Abfrage nur "Poster Paint" zurück.

Dokumente mithilfe von Operatoren abrufen

Die folgende Abfrage gibt alle Dokumente zurück, in denen der Wert "`Inventory.OnHand`" kleiner als 50 ist.

```
db.example.find(
        { "Inventory.OnHand": { $lt: 50 } } )
```

Eine Liste der unterstützten Abfrageoperatoren finden Sie unter Abfrage- und Projektionsoperatoren.

Dokumente werden aktualisiert

In der Regel sind Ihre Dokumente nicht statisch und werden als Teil Ihrer Anwendungs-Workflows aktualisiert. Die folgenden Beispiele zeigen einige der Möglichkeiten, wie Sie Dokumente aktualisieren können.

Um ein bestehendes Dokument zu aktualisieren, verwenden Sie die Operation`update()`. Die Operation `update()` besitzt zwei Dokumentenparameter. Das erste Dokument gibt an, welche Dokumente aktualisiert werden sollen. Das zweite Dokument gibt an, welche Aktualisierungen durchzuführen sind.

Wenn Sie ein vorhandenes Feld aktualisieren — unabhängig davon, ob es sich bei dem Feld um ein einfaches Feld, ein Array oder ein eingebettetes Dokument handelt — geben Sie den Feldnamen und seine Werte an. Am Ende der Operation wirkt es so, als ob das Feld im alten Dokument durch das neue Feld und die Werte ersetzt wurde.

Themen

- Aktualisierung der Werte eines vorhandenen Felds
- Ein neues Feld hinzufügen
- Ein eingebettetes Dokument ersetzen
- Neue Felder in ein eingebettetes Dokument einfügen
- Ein Feld aus einem Dokument entfernen
- Ein Feld aus mehreren Dokumenten entfernen

Aktualisierung der Werte eines vorhandenen Felds

Verwenden Sie die folgenden vier Dokumente, die Sie zuvor hinzugefügt haben, für die folgenden Aktualisierungsoperationen.

```
{
    "Item": "Ruler",
    "Colors": ["Red","Green","Blue","Clear","Yellow"],
    "Inventory": {
        "OnHand": 47,
        "MinOnHand": 40
    },
    "UnitPrice": 0.89
},
{
    "Item": "Pen",
    "Colors": ["Red","Green","Blue","Black"],
    "Inventory": {
        "OnHand": 244,
        "MinOnHand": 72
```

```
    }
},
{
    "Item": "Poster Paint",
    "Colors": ["Red","Green","Blue","Black","White"],
    "Inventory": {
        "OnHand": 47,
        "MinOnHand": 50
    }
},
{
    "Item": "Spray Paint",
    "Colors": ["Black","Red","Green","Blue"],
    "Inventory": {
        "OnHand": 47,
        "MinOnHand": 50,
        "OrderQnty": 36
    }
}
```

So aktualisieren Sie ein einfaches Feld

Um ein einfaches Feld zu aktualisieren, verwenden Sie `update()` mit `$set`, um den Feldnamen und den neuen Wert anzugeben. Im folgenden Beispiel wird das `Item` von "Pen" in "Gel Pen" geändert.

```
db.example.update(
    { "Item" : "Pen" },
    { $set: { "Item": "Gel Pen" } }
)
```

Ergebnisse dieser Operation sehen in etwa folgendermaßen aus.

```
{
    "Item": "Gel Pen",
    "Colors": ["Red","Green","Blue","Black"],
    "Inventory": {
        "OnHand": 244,
        "MinOnHand": 72
    }
}
```

So aktualisieren Sie ein Array

Das folgende Beispiel ersetzt das vorhandene Array von Farben mit einem neuen Array, das in der Liste der Farben `Orange` enthält und `White` weglässt. Die neue Liste von Farben liegt in der Reihenfolge vor, die in der Operation `update()` festgelegt wurde.

```
db.example.update(
    { "Item" : "Poster Paint" },
    { $set: { "Colors": ["Red","Green","Blue","Orange","Black"] } }
)
```

Ergebnisse dieser Operation sehen in etwa folgendermaßen aus.

```
{
    "Item": "Poster Paint",
    "Colors": ["Red","Green","Blue","Orange","Black"],
    "Inventory": {
        "OnHand": 47,
        "MinOnHand": 50
    }
}
```

Ein neues Feld hinzufügen

Um ein Dokument durch Hinzufügen eines oder mehrerer neuer Felder zu ändern, verwenden Sie die Operation `update()` mit einem Abfragedokument, das das einzufügende Dokument und die neuen Felder und Werte, die mit dem Operator `$set` eingefügt werden sollen, identifiziert.

Im folgenden Beispiel wird das Feld `UnitPrice` mit dem Wert `3.99` dem Dokument "Spray Paints" hinzugefügt. Beachten Sie, dass der Wert `3.99` numerisch ist und keine Zeichenfolge.

```
db.example.update(
    { "Item": "Spray Paint" },
    { $set: { "UnitPrice": 3.99 } }
)
```

Die Ergebnisse dieser Operation sehen ungefähr wie folgt aus (JSON-Format).

```
{
    "Item": "Spray Paint",
    "Colors": ["Black","Red","Green","Blue"],
    "Inventory": {
        "OnHand": 47,
```

```
        "MinOnHand": 50,
        "OrderQnty": 36
    },
    "UnitPrice": 3.99
}
```

Ein eingebettetes Dokument ersetzen

Um ein Dokument durch Ersetzen eines eingebetteten Dokuments zu ändern, verwenden Sie die Operation `update()` mit Dokumenten, die das eingebettete Dokument und seine neuen Felder und Werte mit dem Operator `$set` identifizieren.

Anhand des folgenden Dokuments

```
db.example.insert({
    "DocName": "Document 1",
    "Date": {
        "Year": 1987,
        "Month": 4,
        "Day": 18
    }
})
```

So ersetzen Sie ein eingebettetes Dokument

Im folgenden Beispiel wird das aktuellen Datumsdokument durch ein neues ersetzt, das nur die Felder `Month` und `Day` besitzt; `Year` wurde entfernt.

```
db.example.update(
    { "DocName" : "Document 1" },
    { $set: { "Date": { "Month": 4, "Day": 18 } } }
)
```

Ergebnisse dieser Operation sehen in etwa folgendermaßen aus.

```
{
    "DocName": "Document 1",
    "Date": {
        "Month": 4,
        "Day": 18
    }
```

```
}
```

Neue Felder in ein eingebettetes Dokument einfügen

So fügen Sie einem eingebetteten Dokument neue Felder hinzu

Um ein Dokument durch Hinzufügen eines oder mehrerer neuer Felder zu einem eingebetteten Dokument zu ändern, verwenden Sie die Operation `update()` mit Dokumenten, die das eingebettete Dokument identifizieren, und "Punktnotation" zum Angeben des eingebetteten Dokuments und der neuen Felder und Werte verwenden, die mit dem Operator `$set` eingefügt werden sollen.

Beim folgenden Dokument verwendet der folgende Code "Punktnotierung", um die Felder `DoW` und `Year` im eingebetteten `Date`-Dokument und `Words` im übergeordneten Dokument einzufügen.

```
{
    "DocName": "Document 1",
    "Date": {
        "Month": 4,
        "Day": 18
    }
}
```

```
db.example.update(
    { "DocName" : "Document 1" },
    { $set: { "Date.Year": 1987,
              "Date.DoW": "Saturday",
              "Words": 2482 } }
)
```

Ergebnisse dieser Operation sehen in etwa folgendermaßen aus.

```
{
    "DocName": "Document 1",
    "Date": {
        "Month": 4,
        "Day": 18,
        "Year": 1987,
        "DoW": "Saturday"
    },
    "Words": 2482
}
```

Ein Feld aus einem Dokument entfernen

Um ein Dokument zu ändern, indem Sie ein Feld aus dem Dokument entfernen, verwenden Sie die Operation `update()` mit einem Abfragedokument, das das Dokument identifiziert, aus dem das Feld entfernt werden soll, und den Operator `$unset`, um das zu entfernende Feld anzugeben.

Das folgende Beispiel entfernt das Feld `Words` aus dem vorangegangen Dokument.

```
db.example.update(
    { "DocName" : "Document 1" },
    { $unset: { Words:1 } }
)
```

Ergebnisse dieser Operation sehen in etwa folgendermaßen aus.

```
{
    "DocName": "Document 1",
    "Date": {
        "Month": 4,
        "Day": 18,
        "Year": 1987,
        "DoW": "Saturday"
    }
}
```

Ein Feld aus mehreren Dokumenten entfernen

Um ein Dokument zu ändern, indem ein Feld aus mehreren Dokumenten entfernt wird, verwenden Sie die Operation `update()` mit dem Operator `$unset` und der Option `multi`, die auf `true` festgelegt ist.

Im folgenden Beispiel wird das Feld `Inventory` aus allen Dokumenten in der Beispielsammlung entfernt. Wenn ein Dokument das Feld `Inventory` nicht hat, wird keine Aktion für dieses Dokument durchgeführt. Wenn `multi: true` weggelassen wird, wird die Aktion nur für das erste Dokument ausgeführt, das das Kriterium erfüllt.

```
db.example.update(
    {},
    { $unset: { Inventory:1 } },
    { multi: true }
```

```
)
```

Dokumente löschen

Um ein Dokument aus Ihrer Datenbank zu entfernen, verwenden Sie die Operation `remove()` und geben Sie an, welches Dokument Sie entfernen möchten. Der folgende Code entfernt "Gel Pen" aus Ihrer `example`-Sammlung.

```
db.example.remove( { "Item": "Gel Pen" } )
```

Um alle Dokumente aus Ihrer Datenbank zu entfernen, verwenden Sie den `remove()` Vorgang mit einer leeren Abfrage.

```
db.example.remove( { } )
```

Erste Schritte mit Amazon DocumentDB

Es gibt viele Möglichkeiten, eine Verbindung herzustellen und mit Amazon DocumentDB zu beginnen. Dieses Handbuch ist der schnellste, einfachste und einfachste Weg für Benutzer, mit unserer leistungsstarken Dokumentendatenbank zu beginnen. In diesem Handbuch AWS CloudShell können Sie Ihren Amazon DocumentDB-Cluster direkt von der aus verbinden und abfragen. AWS-Managementkonsole Neukunden, die für das AWS kostenlose Kontingent in Frage kommen, können Amazon DocumentDB CloudShell kostenlos nutzen. Wenn Ihre AWS CloudShell Umgebung oder Ihr Amazon DocumentDB-Cluster Ressourcen nutzt, die über das kostenlose Kontingent hinausgehen, werden Ihnen die normalen AWS Tarife für diese Ressourcen berechnet. Mit diesem Handbuch können Sie in weniger als fünf Minuten mit Amazon DocumentDB loslegen.

> **Note**
>
> Die Anweisungen in diesem Handbuch beziehen sich speziell auf die Erstellung und Verbindung zu instanzbasierten Amazon DocumentDB-Clustern, in denen Amazon DocumentDB und verfügbar sind. AWS CloudShell
>
> - Wenn Sie elastische Amazon DocumentDB-Cluster erstellen und eine Verbindung zu diesen herstellen möchten, finden Sie unter. Erste Schritte mit Amazon DocumentDB Elastic Clusters
> - Wenn Sie in den Regionen AWS Chinas ansässig sind, finden Sie weitere Informationen unterStellen Sie automatisch eine Verbindung zu Amazon EC2 her.

Themen

- Voraussetzungen
- Schritt 1: Erstellen eines Clusters
- Schritt 2: Stellen Sie eine Verbindung zu Ihrem Cluster her
- Schritt 3: Daten einfügen und abfragen
- Schritt 4: Erkunden
- Amazon DocumentDB: Schnellstart mit CloudFormation

Voraussetzungen

Bevor Sie Ihren ersten Amazon DocumentDB-Cluster erstellen, müssen Sie Folgendes tun:

Erstellen Sie ein Amazon Web Services (AWS) -Konto

Bevor Sie Amazon DocumentDB verwenden können, müssen Sie über ein Amazon Web Services (AWS) -Konto verfügen. Das AWS Konto ist kostenlos. Sie zahlen nur für die Services und Ressourcen, die Sie wirklich nutzen.

Wenn Sie noch keines haben AWS-Konto, führen Sie die folgenden Schritte aus, um eines zu erstellen.

Um sich für eine anzumelden AWS-Konto

1. Offen https://signin.aws.amazon.com/signup? request_type=registrieren.
2. Folgen Sie den Online-Anweisungen.

   Während der Anmeldung erhalten Sie einen Telefonanruf oder eine Textnachricht und müssen einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für eine registrieren AWS-Konto, wird eine erstellt. Root-Benutzer des AWS-Kontos Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Benutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

Richten Sie die erforderlichen AWS Identity and Access Management (IAM-) Berechtigungen ein.

Für den Zugriff auf die Verwaltung von Amazon DocumentDB-Ressourcen wie Clustern, Instances und Cluster-Parametergruppen sind Anmeldeinformationen erforderlich, mit denen Sie Ihre Anfragen authentifizieren AWS können. Weitere Informationen finden Sie unter Identitäts- und Zugriffsmanagement für Amazon DocumentDB.

1. Geben Sie in der Suchleiste von IAM ein **`IAM`** und wählen Sie im daraufhin angezeigten Drop-down-Menü die Option aus. AWS-Managementkonsole
2. Sobald Sie sich in der IAM-Konsole befinden, wählen Sie im Navigationsbereich Benutzer aus.
3. Wählen Sie Ihren Nutzernamen aus.

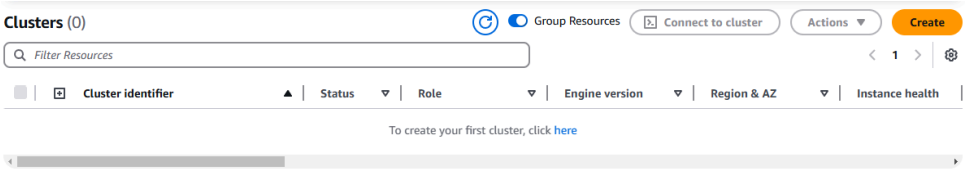
4. Wählen Sie Add permissions (Berechtigungen hinzufügen) aus.
5. Wählen Sie Richtlinien direkt anhängen aus.
6. Geben Sie es `AmazonDocDBFullAccess` in die Suchleiste ein und wählen Sie es aus, sobald es in den Suchergebnissen erscheint.
7. Wählen Sie Weiter aus.
8. Wählen Sie Add permissions (Berechtigungen hinzufügen) aus.

> **Note**
>
> Ihr AWS Konto beinhaltet eine Standard-VPC in jeder Region. Wenn Sie sich für die Verwendung einer Amazon VPC entscheiden, führen Sie die Schritte im Thema Erstellen einer Amazon VPC im Amazon VPC-Benutzerhandbuch aus.

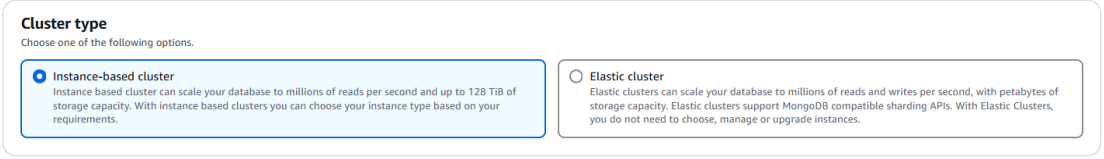
Schritt 1: Erstellen eines Clusters

In diesem Schritt erstellen Sie einen Amazon DocumentDB-Cluster.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Wählen Sie in der Amazon DocumentDB-Managementkonsole unter Clusters die Option Create aus.



3. Wählen Sie auf der Seite „Amazon DocumentDB-Cluster erstellen“ im Abschnitt Cluster-Typ die Option Instance-based Cluster aus (dies ist die Standardoption).

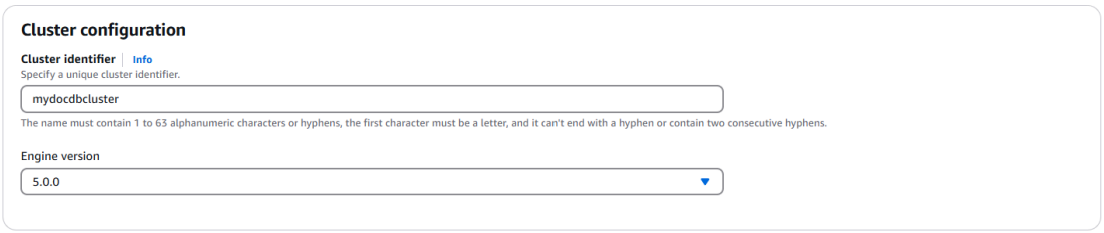


> **Note**
>
> Die andere Option in dieser Kategorie ist Elastic Cluster. Weitere Informationen zu den elastischen Clustern von Amazon DocumentDB finden Sie unter Verwenden von Amazon DocumentDB Elastic Clusters

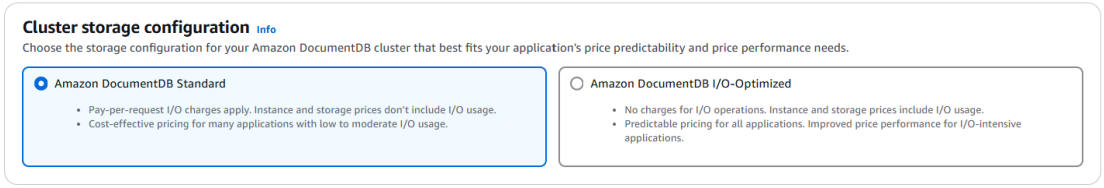
4. Im Abschnitt Cluster-Konfiguration:

   a. Geben Sie als Cluster-ID einen eindeutigen Namen ein, z. **`mydocdbcluster`** B. Beachten Sie, dass die Konsole alle Clusternamen in Kleinbuchstaben umwandelt, unabhängig davon, wie sie eingegeben werden.

   b. Wählen Sie für Engine-Version 5.0.0 aus.



5. Wählen Sie im Abschnitt Cluster-Speicherkonfiguration Amazon DocumentDB Standard aus (dies ist die Standardoption).



> **Note**
>
> Die andere Option in dieser Kategorie ist Amazon I/O-Optimized DocumentDB. Weitere Informationen zu beiden Optionen finden Sie unter Amazon DocumentDB-Cluster-Speicherkonfigurationen

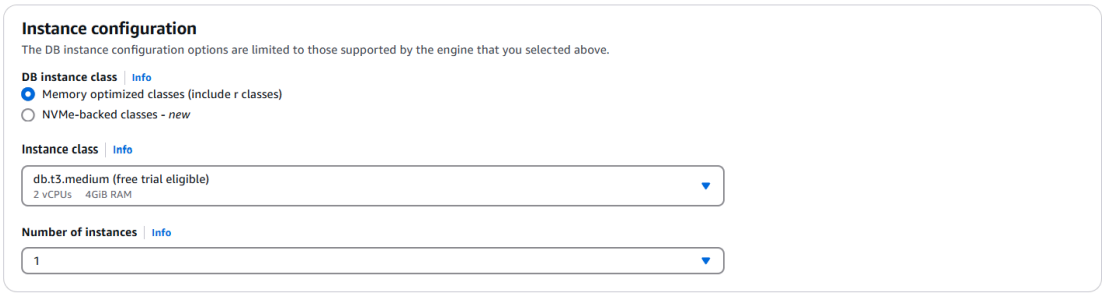
6. Im Abschnitt Instanzkonfiguration:

   a. Wählen Sie für die DB-Instance-Klasse die Option Speicheroptimierte Klassen (einschließlich R-Klassen) aus (dies ist die Standardeinstellung).

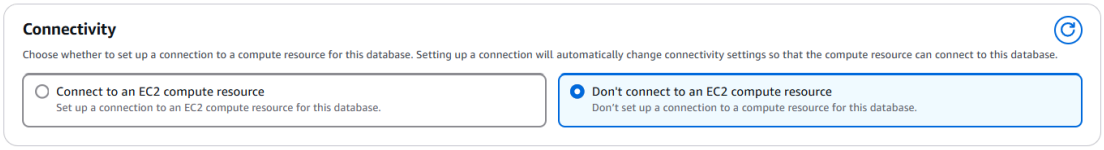
Die andere Instanzoption sind NVMe-backed Klassen. Weitere Informationen hierzu finden Sie unter NVMe-backed Instanzen.

b. Wählen Sie für Instance-Klasse db.t3.medium aus. Dies ist für die kostenlose Testversion berechtigt. AWS

c. Wählen Sie für Anzahl der Instanzen 1 Instance aus. Die Auswahl einer Instanz hilft, die Kosten zu minimieren. Wenn es sich um ein Produktionssystem handeln würde, würden wir empfehlen, drei Instances bereitzustellen, um eine hohe Verfügbarkeit zu gewährleisten.

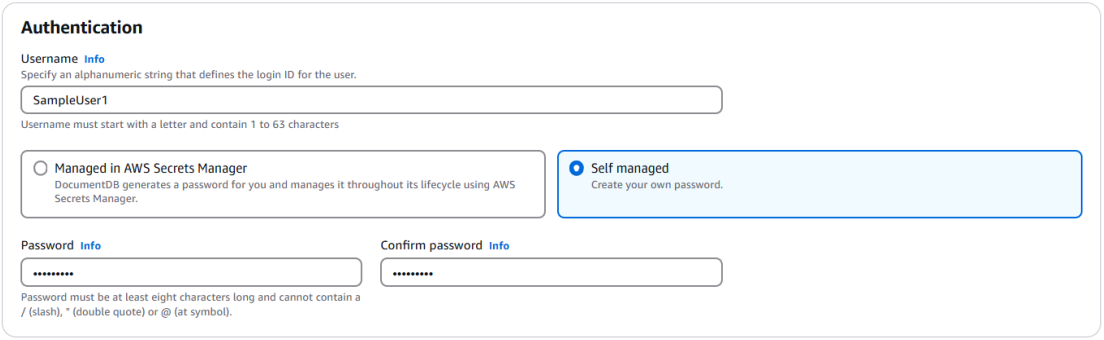


7. Belassen Sie im Abschnitt Konnektivität die Standardeinstellung Keine Verbindung zu einer EC2-Rechenressource herstellen.



8. Geben Sie im Abschnitt Authentifizierung einen Benutzernamen für den Hauptbenutzer ein und wählen Sie dann Selbstverwaltet aus. Geben Sie ein Passwort ein und bestätigen Sie es.

   Wenn Sie stattdessen Verwaltet in ausgewählt haben AWS Secrets Manager, finden Sie Passwortverwaltung mit Amazon DocumentDB und AWS Secrets Manager weitere Informationen unter.



9. Belassen Sie alle anderen Optionen als Standard und wählen Sie Cluster erstellen.

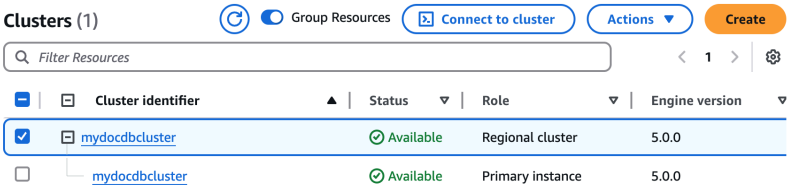
Amazon DocumentDB stellt jetzt Ihren Cluster bereit. Die Fertigstellung kann bis zu ein paar Minuten dauern.

> **Note**
>
> Informationen zu Cluster-Statuswerten finden Sie Werte für den Cluster-Status im Kapitel Überwachung von Amazon DocumentDB.

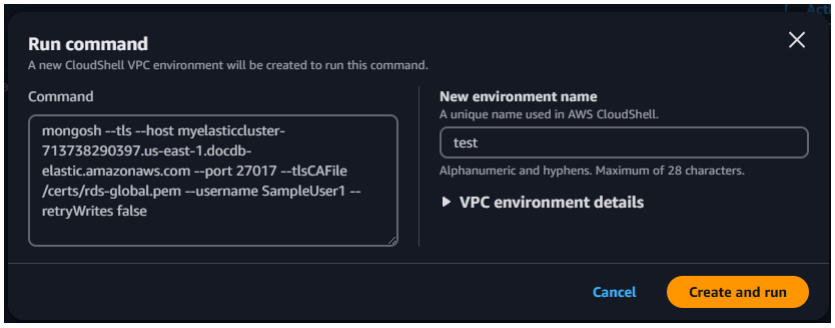
Schritt 2: Stellen Sie eine Verbindung zu Ihrem Cluster her

Stellen Sie eine Verbindung zu Ihrem Amazon DocumentDB-Cluster her mit AWS CloudShell.

1. Suchen Sie in der Amazon-DocumentDB-Verwaltungskonsole unter Cluster den von Ihnen erstellten Cluster. Wählen Sie Ihren Cluster aus, indem Sie das Kontrollkästchen daneben aktivieren.



2. Wählen Sie Mit Cluster verbinden (dies befindet sich neben dem Dropdownmenü Aktionen). Diese Schaltfläche ist erst aktiviert, nachdem Sie auf das Kontrollkästchen neben Ihrem Cluster geklickt haben und der Status sowohl des regionalen Clusters als auch der primären Instance (s) als Verfügbar angezeigt wird. Der Bildschirm „Befehl CloudShell ausführen“ wird angezeigt.
3. Geben Sie im Feld Neuer Umgebungsname einen eindeutigen Namen ein, z. B. „Test“, und wählen Sie Erstellen und ausführen aus. Die VPC-Umgebungsdetails werden automatisch für Ihre Amazon DocumentDB-Datenbank konfiguriert.



4. Wenn Sie dazu aufgefordert werden, geben Sie das Passwort ein, das Sie in Schritt 1: Erstellen Sie einen Amazon DocumentDB-Cluster (Unterschritt 7) erstellt haben.



Nachdem Sie Ihr Passwort eingegeben haben und Ihre Aufforderung erscheint`rs0 [direct: primary] <env-name>>`, sind Sie erfolgreich mit Ihrem Amazon DocumentDB-Cluster verbunden.

> **Note**
>
> Informationen zur Fehlerbehebung finden Sie unter Problembehandlung bei Amazon DocumentDB.

Schritt 3: Daten einfügen und abfragen

Jetzt, da Sie mit Ihrem Cluster verbunden sind, können Sie einige Abfragen ausführen, um sich mit der Verwendung einer Dokumentdatenbank vertraut zu machen.

1. Um ein einzelnes Dokument einzufügen, geben Sie Folgendes ein:

   ```
   db.collection.insertOne({"hello":"DocumentDB"})
   ```

   Sie erhalten die folgende Ausgabe:

   ```
   {
     acknowledged: true,
     insertedId: ObjectId('673657216bdf6258466b128c')
   }
   ```

2. Sie können das Dokument lesen, das Sie mit dem `findOne()` Befehl geschrieben haben (da nur ein einziges Dokument zurückgegeben wird). Geben Sie Folgendes ein:

   ```
   db.collection.findOne()
   ```

Sie erhalten die folgende Ausgabe:

```
{ "_id" : ObjectId("5e401fe56056fda7321fbd67"), "hello" : "DocumentDB" }
```

3. Um ein paar weitere Abfragen durchzuführen, ziehen Sie einen Anwendungsfall mit Spielprofilen in Betracht. Fügen Sie zunächst einige Einträge in eine Sammlung mit dem Titel ein`profiles`. Geben Sie Folgendes ein:

```
db.profiles.insertMany([{ _id: 1, name: 'Matt', status: 'active', level: 12, score:
 202 },
      { _id: 2, name: 'Frank', status: 'inactive', level: 2, score: 9 },
      { _id: 3, name: 'Karen', status: 'active', level: 7, score: 87 },
      { _id: 4, name: 'Katie', status: 'active', level: 3, score: 27 }
])
```

Sie erhalten die folgende Ausgabe:

```
{ acknowledged: true, insertedIds: { '0': 1, '1': 2, '2': 3, '3': 4 } }
```

4. Verwenden Sie den `find()` Befehl, um alle Dokumente in der Profilsammlung zurückzugeben. Geben Sie Folgendes ein:

```
db.profiles.find()
```

Sie erhalten eine Ausgabe, die den Daten entspricht, die Sie in Schritt 3 eingegeben haben.

5. Verwenden Sie mithilfe eines Filters eine Abfrage für ein einzelnes Dokument. Geben Sie Folgendes ein:

```
db.profiles.find({name: "Katie"})
```

Sie erhalten die folgende Ausgabe:

```
{ "_id" : 4, "name" : "Katie", "status": "active", "level": 3, "score":27}
```

6. Versuchen wir nun, ein Profil zu finden und es mit dem `findAndModify` Befehl zu ändern. Wir geben dem Benutzer Matt zusätzliche 10 Punkte mit dem folgenden Code:

```
db.profiles.findAndModify({
```

```
        query: { name: "Matt", status: "active"},
        update: { $inc: { score: 10 } }
    })
```

Du erhältst die folgende Ausgabe (beachte, dass seine Punktzahl noch nicht gestiegen ist):

```
{
    [{_id : 1, name : 'Matt', status: 'active', level: 12, score: 202}]
```

7. Mit der folgenden Abfrage können Sie überprüfen, ob sich sein Ergebnis geändert hat:

   ```
   db.profiles.find({name: "Matt"})
   ```

   Sie erhalten die folgende Ausgabe:

   ```
   { "_id" : 1, "name" : "Matt", "status" : "active", "level" : 12, "score" : 212 }
   ```

Schritt 4: Erkunden

Herzlichen Glückwunsch! Sie haben das Handbuch Erste Schritte für instanzbasierte Amazon DocumentDB-Cluster erfolgreich abgeschlossen.

Was kommt als Nächstes? Erfahren Sie, wie Sie diese Datenbank mit einigen ihrer beliebten Funktionen voll ausschöpfen können:

- Amazon DocumentDB verwalten
- Skalierung
- Sichern und Wiederherstellen

> **Note**
>
> Für den Cluster, den Sie im Rahmen dieser ersten Schritte erstellt haben, fallen weiterhin Kosten an, sofern Sie ihn nicht löschen. Eine Anleitung finden Sie unter Löschen eines Amazon DocumentDB-Clusters.

Amazon DocumentDB: Schnellstart mit CloudFormation

Dieser Abschnitt enthält Schritte und weitere Informationen, die Ihnen den schnellen Einstieg in die Verwendung von Amazon DocumentDB (mit MongoDB-Kompatibilität) erleichtern. CloudFormation Allgemeine Informationen zu Amazon DocumentDB finden Sie unter. Was ist Amazon DocumentDB (mit MongoDB-Kompatibilität)

Diese Anweisungen verwenden eine AWS CloudFormation Vorlage, um einen Cluster und Instances in Ihrer Standard-Amazon-VPC zu erstellen. Anweisungen zum Erstellen dieser Ressourcen finden Sie unter Erste Schritte mit Amazon DocumentDB.

> **Important**
>
> Der CloudFormation Stack, der mit dieser Vorlage erstellt wird, erstellt mehrere Ressourcen, einschließlich Ressourcen in Amazon DocumentDB (z. B. ein Cluster und Instances) und Amazon Elastic Compute Cloud (z. B. eine Subnetzgruppe).
> Einige dieser Ressourcen sind nicht im kostenlosen Kontingent enthalten. Preisinformationen finden Sie unter Amazon DocumentDB-Preise und Amazon EC2-Preise. Sie können den Stack löschen, wenn Sie ihn nicht mehr benötigen, um Gebühren zu sparen.

Dieser CloudFormation Stack ist nur für Tutorialzwecke gedacht. Wenn Sie diese Vorlage für eine Produktionsumgebung verwenden, verwenden Sie strengere IAM-Richtlinien und Sicherheitsmaßnahmen. Informationen zum Sichern von Ressourcen finden Sie unter Amazon VPC Security und Amazon EC2 Network and Security.

Themen

- Voraussetzungen
- Eine Amazon DocumentDB starten CloudFormation Stack
- Zugreifen auf den Amazon DocumentDB-Cluster
- Kündigungsschutz und Löschschutz

Voraussetzungen

Bevor Sie einen Amazon DocumentDB-Cluster erstellen, müssen Sie über Folgendes verfügen:

- Eine standardmäßige Amazon-VPC

- Die erforderlichen IAM-Berechtigungen

Erforderliche IAM-Berechtigungen

Die folgenden Berechtigungen erlauben Ihnen das Erstellen von Ressourcen für den CloudFormation -Stack:

AWS Verwaltete Richtlinien

- `AWSCloudFormationReadOnlyAccess`
- `AmazonDocDBFullAccess`

Zusätzliche IAM-Berechtigungen

Die folgende Richtlinie beschreibt die zusätzlichen Berechtigungen, die zum Erstellen und Löschen dieses CloudFormation Stacks erforderlich sind.

Ersetzen Sie in den folgenden Beispielen jedes *`user input placeholder`* durch die Informationen Ihrer Ressource.

Amazon-EC2-Schlüsselpaar

In der Region, in der Sie den CloudFormation Stack erstellen werden, muss ein Schlüsselpaar (und die PEM-Datei) verfügbar sein. Wenn Sie ein Schlüsselpaar erstellen müssen, finden Sie weitere Informationen unter Erstellen eines Schlüsselpaars mit Amazon EC2 im Amazon EC2-Benutzerhandbuch.

Eine Amazon DocumentDB starten CloudFormation Stack

In diesem Abschnitt wird beschrieben, wie Sie einen Amazon CloudFormation DocumentDB-Stack starten und konfigurieren.

1. Melden Sie sich AWS-Managementkonsole unter an. https://console.aws.amazon.com/
2. In der folgenden Tabelle sind die jeweiligen AWS-Region Amazon DocumentDB-Stack-Vorlagen aufgeführt. Wählen Sie Launch Stack für das AWS-Region , in dem Sie Ihren Stack starten möchten.

Region	Vorlage anzeigen	In Designer anzeigen	Starten
USA Ost (Ohio)	Vorlage anzeigen	In Designer anzeigen	
USA Ost (Nord-Virginia)	Vorlage anzeigen	In Designer anzeigen	
USA West (Oregon)	Vorlage anzeigen	In Designer anzeigen	
Asien-Pazifik (Mumbai)	Vorlage anzeigen	In Designer anzeigen	
Asien-Pazifik (Seoul)	Vorlage anzeigen	In Designer anzeigen	
Asien-Pazifik (Singapur)	Vorlage anzeigen	In Designer anzeigen	
Asien-Pazifik (Sydney)	Vorlage anzeigen	In Designer anzeigen	
Asien-Pazifik (Tokio)	Vorlage anzeigen	In Designer anzeigen	
Kanada (Zentral)	Vorlage anzeigen	In Designer anzeigen	
Europa (Frankfurt)	Vorlage anzeigen	In Designer anzeigen	
Europa (Irland)	Vorlage anzeigen	In Designer anzeigen	
Europa (London)	Vorlage anzeigen	In Designer anzeigen	
Europa (Paris)	Vorlage anzeigen	In Designer anzeigen	

3. Stack erstellen — Beschreibt die Amazon DocumentDB-Vorlage, die Sie ausgewählt haben. Jeder Stack basiert auf einer Vorlage — einer JSON- oder YAML-Datei —, die die Konfiguration

der AWS Ressourcen enthält, die Sie in den Stack aufnehmen möchten. Da Sie sich dafür entschieden haben, einen Stack aus den oben bereitgestellten Vorlagen zu starten, wurde Ihre Vorlage bereits so konfiguriert, dass ein Amazon DocumentDB-Stack für den von AWS-Region Ihnen ausgewählten Stack erstellt wird.

Wenn Sie einen CloudFormation Stack starten, ist der Löschschutz für Ihren Amazon DocumentDB-Cluster standardmäßig deaktiviert. Wenn Sie den Löschschutz für Ihren Cluster aktivieren möchten, führen Sie die folgenden Schritte aus. Andernfalls wählen Sie Next (Weiter), um mit dem nächsten Schritt fortzufahren.

So aktivieren Sie den Löschschutz für Ihren Amazon DocumentDB-Cluster:

1. Wählen Sie in der unteren rechten Ecke der Seite „Stack erstellen“ die Option In Designer anzeigen aus.
2. Ändern Sie die Vorlage mithilfe des integrierten JSON- und YAML-Editors auf der resultierenden CloudFormation Designer-Seite der Konsole. Blättern Sie zum Abschnitt `Resources` und ändern Sie diesen wie folgt, um den `DeletionProtection` einzubinden. Weitere Informationen zur Verwendung von CloudFormation Designer finden Sie unter Was ist CloudFormation Designer? .

   JSON:

```
"Resources": {
    "DBCluster": {
        "Type": "AWS::DocDB::DBCluster",
        "DeletionPolicy": "Delete",
        "Properties": {
            "DBClusterIdentifier": {
                "Ref": "DBClusterName"
            },
            "MasterUsername": {
                "Ref": "MasterUser"
            },
            "MasterUserPassword": {
                "Ref": "MasterPassword"
            },
            "DeletionProtection": "true"
        }
    },
```

YAML:

```
Resources:
  DBCluster:
    Type: 'AWS::DocDB::DBCluster'
    DeletionPolicy: Delete
    Properties:
      DBClusterIdentifier: !Ref DBClusterName
      MasterUsername: !Ref MasterUser
      MasterUserPassword: !Ref MasterPassword
      DeletionProtection: 'true'
```

3. Wählen Sie Create Stack (Stack erstellen)
   ()
   in der oberen linken Ecke der Seite, um Ihre Änderungen zu speichern und einen Stack zu erstellen, bei dem diese Änderungen aktiviert sind.
4. Nachdem Sie Ihre Änderungen gespeichert haben, werden Sie zur Seite Create stack (Stack erstellen) weitergeleitet.
5. Wählen Sie Next (Weiter), um fortzufahren.

4. Stackdetails angeben — Geben Sie den Stacknamen und die Parameter für Ihre Vorlage ein. Parameter werden in Ihrer Vorlage definiert und ermöglichen Ihnen, benutzerdefinierte Werte einzugeben, wenn Sie einen Stack erstellen oder aktualisieren.
   - Geben Sie unter Stack name (Stack-Name) einen Namen für Ihren Stack ein oder übernehmen Sie den angegebenen Namen. Der Stackname kann Buchstaben (A—Z und a—z), Zahlen (0—9) und Bindestriche (—) enthalten.
   - Geben Sie unter Parameter, die folgenden Details ein:
     - DBClusterName— Geben Sie einen Namen für Ihren Amazon DocumentDB-Cluster ein oder akzeptieren Sie den angegebenen Namen.

       Einschränkungen bei der Benennung von Clustern:
       - Die Länge besteht aus [1—63] Buchstaben, Zahlen oder Bindestrichen.
       - Muss mit einem Buchstaben beginnen.
       - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.

- Muss für alle Cluster in Amazon RDS, Neptune und Amazon DocumentDB pro Region eindeutig sein. AWS-Konto

- DBInstanceClass— Wählen Sie aus der Dropdownliste die Instance-Klasse für Ihren Amazon DocumentDB-Cluster aus.
- DBInstanceName— Geben Sie einen Namen für Ihre Amazon DocumentDB-Instance ein oder akzeptieren Sie den angegebenen Namen.

  Einschränkungen für Instance-Benennungen:

  - Die Länge besteht aus [1—63] Buchstaben, Zahlen oder Bindestrichen.
  - Muss mit einem Buchstaben beginnen.
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.
  - Muss für alle Instances in Amazon RDS, Neptune und Amazon DocumentDB pro Region eindeutig sein. AWS-Konto
- MasterPassword— Das Passwort für das Datenbank-Administratorkonto.
- MasterUser— Der Benutzername des Datenbankadministratorkontos. Der MasterUser muss mit einem Buchstaben beginnen und darf nur alphanumerische Zeichen enthalten.

Klicken Sie auf Next (Weiter), um Ihre Änderungen zu speichern und fortzufahren.

5. Stack-Optionen konfigurieren — Konfiguriere die Tags, Berechtigungen und zusätzlichen Optionen deines Stacks.

   - Tags — Geben Sie Tag-Paare (Schlüssel-Wert) an, die auf Ihre Ressourcen in Ihrem Stack angewendet werden sollen. Sie können bis zu 50 eindeutige Tags für jeden Stack hinzufügen.
   - Berechtigungen — Optional. Wählen Sie eine IAM-Rolle, um explizit zu definieren, wie Ressourcen im Stack erstellt, geändert oder gelöscht werden CloudFormation können. Wenn Sie keine Rolle auswählen, werden Berechtigungen CloudFormation verwendet, die auf Ihren Benutzeranmeldeinformationen basieren. Bevor Sie eine Service-Rolle festlegen, stellen Sie sicher, dass Sie über die Berechtigung zum Weiterleiten (`iam:PassRole`) verfügen. Die `iam:PassRole`-Berechtigung gibt an, welche Rollen Sie verwenden können.

     > **Note**
     >
     > Wenn Sie eine Servicerolle angeben, wird diese Rolle CloudFormation immer für alle Operationen verwendet, die auf diesem Stack ausgeführt werden. Andere Benutzer,

die Berechtigungen zum Ausführen von Vorgängen in diesem Stapel haben, können diese Rolle verwenden, selbst wenn Sie keine Berechtigung zum Weiterleiten haben. Wenn die Rolle Berechtigungen umfasst, die der Benutzer nicht haben sollte, können Sie die Berechtigungen eines Benutzers versehentlich weiterleiten. Stellen Sie sicher, dass die Rolle die geringsten Rechte gewährt.

- Erweiterte Optionen — Sie können die folgenden erweiterten Optionen festlegen:
  - Stack-Richtlinie — Optional. Legt die Ressourcen, fest die Sie vor unbeabsichtigten Aktualisierungen während einer Stack-Aktualisierung schützen möchten. Standardmäßig können alle Ressourcen bei einer Stack-Aktualisierung aktualisiert werden.

    Sie können die Stack-Richtlinie direkt im JSON-Format eingeben oder eine JSON-Datei, die die Stack-Richtlinie enthält, hochladen. Weitere Informationen finden Sie unter Prevent Updates to Stack Resources (Verhindern von Aktualisierungen der Stack-Ressourcen).
  - Rollback-Konfiguration — Optional. Geben Sie CloudWatch Log-Alarme CloudFormation an, die beim Erstellen und Aktualisieren des Stacks überwacht werden sollen. Wenn der Vorgang einen Alarmschwellenwert überschreitet, wird er CloudFormation rückgängig gemacht.
  - Benachrichtigungsoptionen — Optional. Geben Sie Themen für das Simple Notification System (SNS) an.
  - Optionen zur Stack-Erstellung — Optional. Sie können die folgenden Optionen angeben:
    - Rollback bei Ausfall — Ob der Stack zurückgesetzt werden soll oder nicht, wenn die Stack-Erstellung fehlschlägt.
    - Timeout — Die Anzahl der Minuten, bis eine Zeitüberschreitung bei der Stack-Erstellung eintritt.
    - Kündigungsschutz — Verhindert, dass der Stack versehentlich gelöscht wird.

      Note

      CloudFormation Der Kündigungsschutz unterscheidet sich vom Amazon DocumentDB-Konzept des Löschschutzes. Weitere Informationen finden Sie unter Kündigungsschutz und Löschschutz.

Wählen Sie Next (Weiter), um fortzufahren.

6. <stack-name>Überprüfung — Prüfen Sie Ihre Stack-Vorlage, Details und Konfigurationsoptionen. Sie können auch einen Schnellerstellungs-Link unten auf der Seite öffnen, um Stacks mit denselben Grundkonfigurationen wie diesen zu erstellen.

   - Wählen Sie Create (Erstellen), um den Stack zu erstellen.
   - Alternativ können Sie auch Create change set (Änderungssatz erstellen) wählen. Ein Änderungssatz ist eine Vorschau darauf, wie dieser Stack vor seiner Erstellung konfiguriert wird. Auf diese Weise können Sie verschiedene Konfigurationen prüfen, bevor Sie den Änderungssatz ausführen.

Zugreifen auf den Amazon DocumentDB-Cluster

Sobald der CloudFormation Stack abgeschlossen ist, können Sie eine Amazon EC2-Instance verwenden, um eine Verbindung zu Ihrem Amazon DocumentDB-Cluster herzustellen. Informationen zum Herstellen einer Verbindung zu einer Amazon EC2-Instance mithilfe von SSH finden Sie unter Connect to Your Linux Instance im Amazon EC2-Benutzerhandbuch.

Nachdem Sie eine Verbindung hergestellt haben, lesen Sie die folgenden Abschnitte, die Informationen zur Verwendung von Amazon DocumentDB enthalten.

- Schritt 3: Daten einfügen und abfragen
- Verwaltung von Amazon DocumentDB-Clustern und -Instances
- Überwachung und Protokollierung in Amazon DocumentDB

Kündigungsschutz und Löschschutz

Es ist eine bewährte Methode von Amazon DocumentDB, den Löschschutz und den Kündigungsschutz zu aktivieren. CloudFormation Der Kündigungsschutz unterscheidet sich deutlich von der Amazon DocumentDB-Löschschutzfunktion.

- Kündigungsschutz — Sie können verhindern, dass ein Stack versehentlich gelöscht wird, indem Sie den Kündigungsschutz für Ihren CloudFormation Stack aktivieren. Wenn ein Benutzer versucht, einen Stack mit aktiviertem Beendigungsschutz zu löschen, schlägt das Löschen fehl und der Stack bleibt unverändert. Der Kündigungsschutz ist standardmäßig deaktiviert, wenn Sie einen Stack mit erstellen CloudFormation. Sie können den Beendigungsschutz aktivieren, wenn Sie den Stack erstellen. Weitere Informationen finden Sie unter CloudFormation Stack-Optionen festlegen.

- Löschschutz — Amazon DocumentDB bietet auch die Möglichkeit, den Löschschutz für einen Cluster zu aktivieren. Wenn ein Benutzer versucht, einen Amazon DocumentDB-Cluster mit aktiviertem Löschschutz zu löschen, schlägt das Löschen fehl und der Cluster bleibt unverändert. Wenn der Löschschutz aktiviert ist, schützt er vor versehentlichem Löschen aus der Amazon DocumentDB AWS-Managementkonsole, und. AWS CLI CloudFormation Weitere Informationen zum Aktivieren und Deaktivieren des Löschschutzes für einen Amazon DocumentDB-Cluster finden Sie unter. Löschschutz

Migration und Aktualisierung von Amazon DocumentDB

Amazon DocumentDB ist ein vollständig verwalteter Datenbankdienst, der mit der MongoDB-API kompatibel ist. Sie können Ihre Daten aus MongoDB-Datenbanken, die vor Ort oder in Amazon Elastic Compute Cloud (Amazon EC2) laufen, mithilfe des in diesem Abschnitt beschriebenen Verfahrens zu Amazon DocumentDB migrieren.

Themen

- Migrieren Sie zu Amazon DocumentDB mit AWS Database Migration Service (DMS): Schnellstartanleitung
- Amazon DocumentDB-Migrationsrunbook
- Migration vom Couchbase Server
- Unterstützungstermine der Amazon DocumentDB DocumentDB-Engine-Version
- Direktes Upgrade der Hauptversion von Amazon DocumentDB
- Upgrade der Nebenversion von Amazon DocumentDB
- Aktualisieren Ihres Amazon DocumentDB-Clusters mit AWS Database Migration Service
- Long-Term Support- (LTS, Langzeit-Support) Versionen nutzen
- Erweiterter Support für Amazon DocumentDB

Migrieren Sie zu Amazon DocumentDB mit AWS Database Migration Service (DMS): Schnellstartanleitung

Topics

- Bereiten Sie die DMS-Quelle vor
- DMS einrichten
- Aktivieren Sie die DocumentDB-Komprimierung
- Erstellen Sie eine Replikationsaufgabe
- Überwachen des Fortschritts
- Zusätzliche Informationen

Bereiten Sie die DMS-Quelle vor

Weitere Informationen finden Sie unter Change Streams aktivieren So aktivieren Sie DocumentDB-Change-Streams oder aktivieren das MongoDB-Oplog zur Unterstützung von DMS Change Data Capture (CDC).

- Die DMS-Quelle muss alle laufenden Änderungen beibehalten, bis der vollständige DMS-Ladevorgang für alle enthaltenen Sammlungen abgeschlossen ist.
- DocumentDB-Change-Streams sind zeitbasiert. Stellen Sie sicher, dass Ihre `change_stream_log_retention_duration` Einstellung groß genug ist, um die Zeit bis zum vollständigen Laden abzudecken.
- Das MongoDB-Oplog hat eine feste Größe. Stellen Sie sicher, dass es so dimensioniert ist, dass alle Operationen bei voller Last ausgeführt werden können.

DMS einrichten

Erstellen Sie DMS-Instanz, Quell- und Zielendpunkte und testen Sie jeden Endpunkt.

Aktivieren Sie die DocumentDB-Komprimierung

Amazon DocumentDB 5.0: Aktivieren Sie die Komprimierung, indem Sie eine benutzerdefinierte Parametergruppe an Ihren Amazon DocumentDB-Cluster anhängen und den Parameter auf setzen. `default_collection_compression` `enabled` Weitere Informationen finden Sie unter Verwaltung der Dokumentenkomprimierung auf Sammlungsebene.

Amazon DocumentDB 8.0: Die Zstd-Komprimierung ist standardmäßig aktiviert. Weitere Informationen finden Sie unter Verwaltung der wörterbuchbasierten Komprimierung in Amazon DocumentDB 8.0.

Erstellen Sie eine Replikationsaufgabe

1. Wählen Sie in der DMS-Konsole im Navigationsbereich Migrieren oder replizieren und dann Aufgaben aus.
2. Wählen Sie Create task aus.
3. Gehen Sie auf der Seite „Aufgabe erstellen“ im Abschnitt Aufgabenkonfiguration wie folgt vor:
    - Geben Sie eine eindeutige und aussagekräftige Task-ID ein (z. B. „mongodb-docdb-replication“).

- Wählen Sie im Dropdownmenü Quelldatenbank-Endpunkt den Quellendpunkt aus, den Sie zuvor erstellt haben.
- Wählen Sie im Dropdownmenü Zieldatenbankendpunkt den Zielendpunkt aus, den Sie zuvor erstellt haben.
- Wählen Sie als Aufgabentyp die Option Migrieren und replizieren aus.

4. Im Abschnitt „Einstellungen“:

- Markieren Sie für Task-Logs das Kästchen CloudWatch Logs einschalten.
- Wählen Sie für den Bearbeitungsmodus (oben im Abschnitt) den JSON-Editor aus und legen Sie die folgenden Attribute fest:
  - Setzen Sie `ParallelApplyThreads` den Wert auf 5 (unter`TargetMetadata`). Dies ermöglicht ~1000 insert/update /delete Ops pro Sekunde in CDC.
  - Auf 16 (`MaxFullLoadSubTasks`unter) setzen. `FullLoadSettings` Erwägen Sie, diesen Wert je nach Größe Ihrer Instanz zu erhöhen.
  - Aktivieren Sie für große Sammlungen (über 100 GB) die automatische Partitionierung (unter Tabellenzuordnung und unter dem `parallel-load` Attribut):
    - „type“: „Partitions-Auto“
    - „Anzahl der Partitionen“: 16

Überwachen des Fortschritts

Verwenden Sie die AWS DMS Konsole oder erstellen Sie ein benutzerdefiniertes Dashboard (Dashboarder-Tool), um die Migration zu verfolgen. Konzentrieren Sie sich auf die folgenden Kennzahlen:

- FullLoadThroughputBandwidthTarget— Misst die Netzwerkbandbreite (in KB/second), die von DMS bei der Übertragung von Daten an die Zieldatenbank während der Volllastphase der Migration verwendet wird.
- CDCLatencyTarget— Misst die Zeitverzögerung (in Sekunden) zwischen dem Zeitpunkt, an dem eine Änderung in der Quelldatenbank erfolgt, und dem Zeitpunkt, an dem diese Änderung auf die Zieldatenbank angewendet wird.
- CDCThroughputRowsTarget— Misst die Anzahl der Zeilen pro Sekunde, die DMS während der laufenden Replikationsphase der Migration auf die Zieldatenbank anwendet.

Zusätzliche Informationen

Weitere Informationen zu Amazon DocumentDB finden Sie unter AWS DMS:

- Amazon DocumentDB-Migrationsrunbook
- Migration von MongoDB zu Amazon DocumentDB

Amazon DocumentDB-Migrationsrunbook

Dieses Runbook bietet eine umfassende Anleitung für die Migration einer MongoDB-Datenbank zu Amazon DocumentDB mithilfe von (DMS). AWS Database Migration Service Es wurde entwickelt, um Datenbankadministratoren, Cloud-Ingenieure und Entwickler während des gesamten Migrationsprozesses zu unterstützen — von der ersten Erkennung bis zur Validierung nach der Migration.

Angesichts der Unterschiede in der Implementierung und den unterstützten Funktionen zwischen MongoDB und Amazon DocumentDB betont dieses Runbook einen strukturierten und systematischen Ansatz. Es beschreibt wichtige Bewertungen vor der Migration, hebt Kompatibilitätsaspekte hervor und beschreibt die wichtigsten Aufgaben, die für eine erfolgreiche Migration mit minimalen Unterbrechungen erforderlich sind.

Das Runbook ist in die folgenden Themen gegliedert:

- Kompatibilität— Machen Sie sich mit den unterstützten MongoDB-Funktionen und Datentypen in Amazon DocumentDB vertraut und identifizieren Sie potenzielle Inkompatibilitäten.
- Workload-Erkennung— Analysieren Sie bestehende MongoDB-Workloads, einschließlich read/write Muster, Datenmengen und Leistungsbasislinien.
- Index-Migration— Analysieren Sie Strategien zum Extrahieren und Transformieren von MongoDB-Indizes für eine optimale Leistung in Amazon DocumentDB.
- Benutzermigration— Erläutern Sie den Ansatz für die Migration von Datenbankbenutzern, Rollen und Zugriffskontrollen zu Amazon DocumentDB.
- Datenmigrationen— Behandeln Sie verschiedene Methoden für die Datenmigration mithilfe von AWS DMS CDC (Full Load and Change Data Capture).
- Überwachen— Erläutern Sie die verschiedenen Überwachungsansätze bei der Migration mithilfe von DMS oder nativen Tools.

- Validierung— Stellen Sie Verfahren für Datenintegritätsprüfungen, Funktionsvalidierung und Leistungsvergleich nach der Migration bereit.

Indem die Teams die Anweisungen in diesem Runbook befolgen, können sie einen reibungslosen, sicheren und effizienten Übergang zu Amazon DocumentDB sicherstellen und gleichzeitig die Anwendungsfunktionalität beibehalten und Risiken minimieren.

Kompatibilität

Topics

- Kompatibilität der wichtigsten Funktionen
- Tool zur Kompatibilitätsbewertung von Amazon DocumentDB

Bei der Migration von MongoDB zu Amazon DocumentDB sind eine gründliche Erstbewertung und Überprüfung der Funktionskompatibilität für eine erfolgreiche Migration unerlässlich. Dieser Prozess beginnt mit einer umfassenden Bestandsaufnahme Ihrer MongoDB-Funktionen, einschließlich Aggregationspipeline-Operatoren, Abfragemustern, Indizes und Datenmodellen.

Da Amazon DocumentDB mit den MongoDB-APIs 3.6, 4.0, 5.0 und 8.0 kompatibel ist, müssen Anwendungen, die neuere MongoDB-specific Funktionen verwenden, möglicherweise überarbeitet werden. Zu den kritischen Bereichen, die bewertet werden müssen, gehören Sharding-Mechanismen (Amazon DocumentDB verwendet einen anderen Ansatz), Transaktionsimplementierungen, Change-Stream-Funktionen und Indextypen (insbesondere spärliche und partielle Indizes).

Die Leistungsmerkmale unterscheiden sich ebenfalls. Amazon DocumentDB ist für Unternehmens-Workloads mit vorhersehbarer Leistung optimiert. Bei den Tests sollten repräsentative Workloads auf beiden Systemen ausgeführt werden, um Abfragemuster zu identifizieren, die möglicherweise optimiert werden müssen.

Während der Bewertungsphase ist es wichtig, die Ausführungspläne zu überwachen, um potenzielle Leistungslücken zu erkennen. Dies hilft dabei, einen klaren Migrationsplan zu erstellen, notwendige Anwendungsänderungen zu identifizieren und realistische Zeitpläne für einen reibungslosen Übergang festzulegen.

Kompatibilität der wichtigsten Funktionen

Umfassende Unterstützung von Funktionen

- CRUD-Operationen — Genießen Sie die volle Unterstützung für alle grundlegenden Erstellungs-, Lese-, Aktualisierungs- und Löschvorgänge, einschließlich Massen- und Abfrageoperatoren, und sorgen Sie so für eine nahtlose Anwendungskompatibilität.
- Umfassende Indizierungsfunktionen — Nutzen Sie die umfassende Unterstützung für Einzelfeld-, Verbund-, TTL-, Teil-, Sparse- und 2dSphere-Indizes, um Ihre Abfrageleistung und Textindizes (Version 5) für textbasierte Suchvorgänge zu optimieren.
- Enterprise-grade Replikation — Profitieren Sie von einem robusten automatischen Failover-Mechanismus mit Read Replicas für eine überragende Hochverfügbarkeit ohne Betriebsaufwand.
- Fortschrittliche Backup-Lösungen — Mit dem automatisierten Backup-System mit Point-in-Time Recovery (PITR) und manuellen Snapshots auf Abruf zur Datensicherung können Sie sich keine Sorgen machen.

Verbessert AWS-integrierte Funktionen

- Optimierte Aggregation — Nutzen Sie die am häufigsten verwendeten Aggregationsstufen (`$match`, `$group` `$sort``$project`, usw.) mit optimierter Leistung für Unternehmens-Workloads.
- Transaktionsunterstützung — Implementieren Sie Transaktionen mit mehreren Dokumenten und Sammlungen, perfekt für die meisten Geschäftsanwendungen.
- Real-time Datenverfolgung — Aktivieren Sie Change-Streams mit einem einfachen Befehl und erhöhen Sie die Aufbewahrungsdauer von Change-Streams mithilfe einer einfachen Parametergruppeneinstellung für die Überwachung von Datenänderungen in Echtzeit.
- Location-based Dienste — Implementieren Sie Geodatenanwendungen mit Unterstützung für `$geoNear` Operator- und 2dSphere-Indizes.
- Textsuchfunktionen — Nutzen Sie die integrierte Textsuchfunktion für die Suche nach Inhalten.

Vorteile einer modernen Architektur

- Cloud-native Design — Genießen Sie eine AWS optimierte Architektur, die ältere Funktionen ersetzt, z. B. MapReduce durch effizientere Aggregations-Pipeline-Abläufe.
- Verbesserte Sicherheit — Profitieren Sie von AWS Identity and Access Management (IAM),, SCRAM-SHA-1 SCRAM-SHA-256, X.509 Zertifikatsauthentifizierung und kennwortbasierter Authentifizierung.

- Vorhersagbare Leistung — Erleben Sie eine konsistente Leistung, die speziell für Unternehmens-Workloads optimiert wurde.

Einen umfassenden Überblick über die Funktionen von Amazon DocumentDB finden Sie unter Unterstützte MongoDB-APIs, Operationen und Datentypen in Amazon DocumentDB und Funktionale Unterschiede: Amazon DocumentDB und MongoDB zur Maximierung des Potenzials Ihrer Datenbank.

Amazon DocumentDB unterstützt nicht alle von MongoDB angebotenen Indizes. Wir stellen ein kostenloses Index-Tool zur Verfügung, GitHub um die Kompatibilität zu überprüfen. Führen Sie das Index-Tool aus, um die Inkompatibilität zu beurteilen und entsprechende Problemumgehungen zu planen.

Tool zur Kompatibilitätsbewertung von Amazon DocumentDB

Das Amazon DocumentDB-Kompatibilitätstool GitHub analysiert Ihre MongoDB Arbeitslast und meldet Operatoren, Befehle und Funktionen, die nicht unterstützt werden oder Änderungen in Amazon DocumentDB erfordern. Es unterstützt Zielversionen von 3.6 bis 8.0.1 (Standard: 8.0.1).

Empfohlen: URI-Modus (5.0+) MongoDB

Der URI-Modus stellt eine direkte Verbindung zu Ihrer MongoDB-Instance her, tastet Operationen in Echtzeit ab und erfordert keine Profilerstellung oder Protokollzugriff.

Alternative Methoden

- Log-based Analyse — Analysiert MongoDB Profiler-Protokolle. Erfasst das tatsächliche Laufzeitverhalten, erfordert jedoch die Aktivierung der Profilerstellung und deckt nur den Protokollierungszeitraum ab.
- Quellcodeanalyse — Scannt den Quellcode der Anwendung auf MongoDB API-Nutzung. Bietet eine umfassende Abdeckung, kennzeichnet jedoch möglicherweise ungenutzte Codepfade und kann dynamisch erstellte Abfragen nicht erkennen.

Voraussetzungen

- Python3,7+
- Klonen Sie das Tool: `git clone https://github.com/awslabs/amazon-documentdb-tools.git`

- Abhängigkeiten installieren: `pip install -r compat-tool/requirements.txt`

Vollständige Nutzungsdetails und alle verfügbaren Optionen finden Sie in der README-Datei unter GitHub.

Workload-Erkennung

Die Migration von MongoDB zu Amazon DocumentDB erfordert ein gründliches Verständnis der vorhandenen Datenbank-Arbeitslast. Bei der Workload-Erkennung werden Ihre Datenbanknutzungsmuster, Datenstrukturen, Abfrageleistung und betriebliche Abhängigkeiten analysiert, um einen nahtlosen Übergang mit minimalen Unterbrechungen zu gewährleisten. In diesem Abschnitt werden die wichtigsten Schritte zur Workload-Erkennung beschrieben, um eine effektive Migration von MongoDB zu Amazon DocumentDB zu ermöglichen.

Topics

- Bewertung der bestehenden MongoDB-Bereitstellung
- Identifizierung von Unterschieden im Datenmodell
- Abfrage- und Leistungsanalyse
- Überprüfung der Sicherheit und Zugriffskontrolle
- Überlegungen zum Betrieb und zur Überwachung

Bewertung der bestehenden MongoDB-Bereitstellung

Vor der Migration ist es wichtig, die aktuelle MongoDB-Umgebung zu evaluieren, einschließlich:

- Cluster-Architektur — Identifizieren Sie die Anzahl der Knoten, Replikatsätze und Shardingkonfigurationen. Bei der Migration von MongoDB zu Amazon DocumentDB ist es wichtig, Ihre MongoDB-Sharding-Konfiguration zu verstehen, da Amazon DocumentDB kein benutzergesteuertes Sharding unterstützt. Anwendungen, die für eine geteilte MongoDB-Umgebung entwickelt wurden, benötigen architektonische Änderungen, da Amazon DocumentDB mit seiner speicherbasierten Architektur einen anderen Skalierungsansatz verwendet. Bei der Umstellung auf Amazon DocumentDB müssen Sie Ihre Strategie zur Datenverteilung anpassen und möglicherweise gemeinsam genutzte Sammlungen konsolidieren.
- Speicher- und Datenvolumen — Messen Sie die Gesamtdaten- und Indexgröße Ihres Clusters. Ergänzen Sie dies mit dem Oplog-Review-Tool, um Schreibmuster und die Geschwindigkeit des

Datenwachstums zu verstehen. Weitere Informationen zur Dimensionierung Ihres Clusters finden Sie unterDimensionierung der Instanz.

- Arbeitslastmuster — Analysieren Sie den Lese- und Schreibdurchsatz, die Häufigkeit der Abfrageausführung und die Indexierungseffizienz.
- Betriebsabhängigkeiten — Dokumentieren Sie alle Anwendungen, Dienste und Integrationen, die auf MongoDB basieren.

Identifizierung von Unterschieden im Datenmodell

Amazon DocumentDB ist es zwar MongoDB-compatible, es gibt jedoch Unterschiede bei den unterstützten Funktionen, wie z. B.:

- Transaktionen — Amazon DocumentDB unterstützt ACID-Transaktionen, allerdings mit einigen. Einschränkungen
- Schemadesign — Stellen Sie sicher, dass die Dokumentstrukturen, eingebetteten Dokumente und Verweise den Best Practices von Amazon DocumentDB entsprechen.

Abfrage- und Leistungsanalyse

Das Verständnis des Abfrageverhaltens hilft bei der Optimierung der Migration und der Leistung nach der Migration. Zu den wichtigsten zu analysierenden Bereichen gehören:

- Langsame Abfragen — Identifizieren Sie Abfragen mit hoher Ausführungszeit mithilfe der Profiling-Tools von MongoDB.
- Abfragemuster — Kategorisieren Sie gängige Abfragetypen, einschließlich CRUD-Operationen und -Aggregationen.
- Indexnutzung — Beurteilen Sie, ob Indizes in Amazon DocumentDB effektiv genutzt werden oder optimiert werden müssen. Um die Indexnutzung zu beurteilen und die Leistung in Amazon DocumentDB zu optimieren, verwenden Sie die `$indexStats` Aggregations-Pipeline-Phase in Kombination mit der `explain()` Methode für Ihre kritischen Abfragen. Beginnen Sie mit der Ausführung`db.collection.aggregate([{$indexStats{}}])`, um zu ermitteln, welche Indizes verwendet werden. Sie können eine detailliertere Analyse durchführen, indem Sie Ihre häufigsten Abfragen mit `explainPlan` ausführen.
- Parallelität und Verteilung der Arbeitslast — Evaluieren Sie das Lese- und Schreibverhältnis, das Verbindungspooling und Leistungsengpässe.

Überprüfung der Sicherheit und Zugriffskontrolle

Authentifizierung und Autorisierung

- MongoDB RBAC zu Amazon DocumentDB IAM und RBAC — Ordnen Sie die rollenbasierten Benutzer und Rollen für die Zugriffskontrolle von MongoDB den AWS Identity and Access Management (IAM-) Richtlinien und den Benutzern der Amazon DocumentDB SCRAM-Authentifizierung zu.
- Strategie zur Benutzermigration — Planen Sie die Migration von Datenbankbenutzern, benutzerdefinierten Rollen und Rechten zu den von Amazon DocumentDB unterstützten Authentifizierungsmechanismen.
- Rechteunterschiede — Identifizieren Sie MongoDB-Rechte ohne direkte Amazon DocumentDB-Entsprechungen (z. B. Cluster-Administrationsrollen).
- Anwendungsauthentifizierung — Aktualisieren Sie die Verbindungszeichenfolgen und die Verwaltung der Anmeldeinformationen für die Passwortrichtlinien von Amazon DocumentDB. Sie können den Secrets Manager verwenden, um Ihre Anmeldeinformationen zu speichern und Passwörter zu rotieren.
- Verwaltung von Dienstkonten — Richten Sie Prozesse für die Verwaltung der Anmeldeinformationen für Dienstkonten in ein AWS Secrets Manager.
- Implementierung der geringsten Rechte — Überprüfen und verfeinern Sie die Zugriffskontrollen, um die Prinzipien der geringsten Zugriffsrechte in der neuen Umgebung zu implementieren.

Verschlüsselung

Stellen Sie sicher, dass die Verschlüsselung im Ruhezustand und bei der Übertragung den Compliance-Anforderungen entspricht.

Netzwerkkonfiguration

Planen Sie die Einrichtung der Virtual Private Cloud (VPC) und die Regeln für Sicherheitsgruppen ein.

Überlegungen zum Betrieb und zur Überwachung

Um die Systemzuverlässigkeit aufrechtzuerhalten, sollte die Workload-Erkennung auch Folgendes beinhalten:

- Sicherungs- und Wiederherstellungsstrategie — Evaluieren Sie die vorhandenen Sicherungsmethoden und die Sicherungsfunktionen von Amazon DocumentDB.
- AWS Backup Integration — Nutzen Sie AWS Backup das zentrale Backup-Management für AWS Dienste wie Amazon DocumentDB.
- CloudWatch Metriken — Ordnen Sie die MongoDB-Überwachungsmetriken den Amazon CloudWatch DocumentDB-Metriken für CPU, Arbeitsspeicher, Verbindungen und Speicher zu.
- Performance Insights — Implementieren Sie Amazon DocumentDB Performance Insights, um die Datenbanklast zu visualisieren und Leistungsprobleme mithilfe detaillierter Abfrageanalysen zu analysieren.
- Profiler — Konfigurieren Sie den Amazon DocumentDB-Profiler, um langsam laufende Vorgänge zu erfassen (ähnlich dem Profiler von MongoDB, jedoch mit Amazon-Einstellungen). DocumentDB-specific
  - Über Parametergruppen mit entsprechenden Schwellenwerten aktivieren.
  - Analysieren Sie Profiler-Daten, um Optimierungsmöglichkeiten zu identifizieren
- CloudWatch Ereignisse — Richten Sie eine ereignisgesteuerte Überwachung für Amazon DocumentDB-Cluster-Ereignisse ein.
  - Konfigurieren Sie Benachrichtigungen für Backup-Ereignisse, Wartungsfenster und Failovers.
  - Integrieren Sie Amazon SNS für Warnmeldungen und automatische Reaktionen AWS Lambda .
- Auditprotokollierung — Planen Sie die Konfiguration der Auditprotokollierung ein, um Benutzeraktivitäten und sicherheitsrelevante Ereignisse nachzuverfolgen.
- Verbesserte Überwachung — Aktivieren Sie die erweiterte Überwachung granularer OS-level Kennzahlen in Intervallen von 1 Sekunde.

Neuen Zielcluster erstellen

Erstellen Sie einen Amazon DocumentDB-Cluster, indem Sie den Schritten unter folgen. Erstellen eines Amazon DocumentDB-Clusters Beachten Sie bei der Konfiguration des Clusters für die Migration die folgenden Empfehlungen:

- Instanzklasse: Wählen Sie R8G, um die beste Leistung zu erzielen. Wenn R8G in Ihrer Zielregion nicht verfügbar ist, verwenden Sie R6G. Wählen Sie eine so große Instance wie möglich, um den besten Durchsatz bei voller Auslastung zu erzielen. Nach Abschluss der Migration wird die Skalierung heruntergefahren.
- Engine-Version: neueste Version (8.0.1)

- Anzahl der Instanzen: Wählen Sie 1 Instanz aus, um die Kosten während der Migration zu minimieren. Skalieren Sie auf 3 Instanzen, um eine hohe Verfügbarkeit zu gewährleisten, nachdem die Volllast-Migration abgeschlossen ist.
- Netzwerkeinstellungen: Stellen Sie sicher, dass Ihre Amazon DocumentDB-Sicherheitsgruppe eingehende Verbindungen von der Sicherheitsgruppe der DMS-Replikationsinstanz (für Online-Migrationen) oder vom EC2-Migrationshost (für Offline-Migrationen mit Mongorestore) zulässt.

Index-Migration

Bei der Migration von MongoDB zu Amazon DocumentDB werden nicht nur Daten, sondern auch Indizes übertragen, um die Abfrageleistung aufrechtzuerhalten und den Datenbankbetrieb zu optimieren. In diesem Abschnitt wird der detaillierte, schrittweise Prozess zur Migration von Indizes von MongoDB zu Amazon DocumentDB beschrieben und gleichzeitig Kompatibilität und Effizienz gewährleistet.

Verwenden des Amazon DocumentDB-Indextools

Klonen Sie das Index-Tool auf GitHub

```
git clone https://github.com/awslabs/amazon-documentdb-tools.git
cd amazon-documentdb-tools/index-tool
```

```
pip install -r requirements.txt
```

Indizes aus MongoDB exportieren (wenn Sie von MongoDB migrieren)

```
python3 migrationtools/documentdb_index_tool.py \
  --dump-indexes \
  --dir index_export \
  --uri 'mongodb://<username>:<password>@<source_endpoint>:27017'
```

Indizes verifizieren

```
python3 migrationtools/documentdb_index_tool.py \
  --show-issues \
  --dir index_export
```

Indizes importieren

```
python3 migrationtools/documentdb_index_tool.py \
  --restore-indexes \
  --skip-incompatible \
  --dir index_export \
  --uri 'mongodb://<username>:<password>@<target_endpoint>:27017/?
tls=true&tlsCAFile=global-bundle.pem&replicaSet=rs0&retryWrites=false'
```

Benutzermigration

Die Migration von Benutzern von MongoDB zu Amazon DocumentDB ist für die Aufrechterhaltung der Zugriffskontrolle, Authentifizierung und Datenbanksicherheit unerlässlich. In diesem Abschnitt werden detaillierte Schritte zur erfolgreichen Migration von MongoDB-Benutzern unter Beibehaltung ihrer Rollen und Berechtigungen mithilfe des Amazon DocumentDB-Export-Benutzertools beschrieben.

Verwenden des Amazon DocumentDB-Tools zum Exportieren von Benutzern

Das `Export Users tool` exportiert Benutzer und Rollen aus MongoDB oder Amazon DocumentDB in JavaScript Dateien, die dann verwendet werden können, um sie in einem anderen Cluster neu zu erstellen.

Voraussetzungen

```
# Clone the repository
git clone https://github.com/awslabs/amazon-documentdb-tools.git
cd amazon-documentdb-tools/migration/export-users
```

```
# Install required dependencies
pip install pymongo
```

Schritt 1: Benutzer und Rollen exportieren

```
# Export users and roles to JavaScript files
python3 docdbExportUsers.py \
  --users-file mongodb-users.js \
  --roles-file mongodb-roles.js \
  --uri "mongodb://<username>:<password>@<source_endpoint>:27017"
```

Schritt 2: Bearbeiten Sie die Benutzerdatei

Passwörter werden aus Sicherheitsgründen nicht exportiert. Öffnen Sie die `mongodb-users.js` Datei und fügen Sie für jeden Benutzer ein Passwort hinzu, indem Sie den pwd Wert in jeder `createUser` Anweisung ersetzen:

```
db.getSiblingDB("admin").createUser({
  user: "appuser",
  pwd: "REPLACE_THIS_PASS",
  roles: [
    { role: "readWrite", db: "mydb" }
  ]
})
```

Schritt 3: Stellen Sie benutzerdefinierte Rollen in Amazon DocumentDB wieder her

```
# Import roles first
mongosh \
  --tls \
  --host <target_endpoint>:27017 \
  --tlsCAFile global-bundle.pem \
  --username <username> \
  --password <password> \
  mongodb-roles.js
```

Schritt 4: Benutzer in Amazon DocumentDB wiederherstellen

```
# Import users after roles are created
mongosh \
  --tls \
  --host <target_endpoint>:27017 \
  --tlsCAFile global-bundle.pem \
  --username <username> \
  --password <password> \
  mongodb-users.js
```

Wichtige Hinweise

- Passwörter werden aus Sicherheitsgründen nicht exportiert und müssen manuell zur Datei users.js hinzugefügt werden.
- Rollen müssen vor Benutzern importiert werden, um eine korrekte Rollenzuweisung zu gewährleisten.

- Das Tool generiert JavaScript Dateien, die direkt mit der Mongosh-Shell ausgeführt werden können.
- Benutzerdefinierte Rollen und ihre Rechte bleiben während der Migration erhalten.
- Dieser Ansatz ermöglicht die Überprüfung und Änderung der Benutzerberechtigungen vor dem Import.

Diese Methode bietet einen sicheren und flexiblen Ansatz für die Migration von Benutzern und Rollen von MongoDB zu Amazon DocumentDB und ermöglicht gleichzeitig das Zurücksetzen von Passwörtern während des Migrationsprozesses.

Datenmigrationen

Topics

- Online-Migration
- Offline-Migration

Online-Migration

In diesem Abschnitt finden Sie detaillierte Schritte zur Durchführung einer Online-Migration von MongoDB zu Amazon DocumentDB, um minimale Ausfallzeiten und eine kontinuierliche Replikation AWS DMS zu ermöglichen. Zunächst richten Sie einen Amazon DocumentDB-Cluster als Ziel ein und stellen sicher, dass Ihre MongoDB-Instance ordnungsgemäß als Quelle konfiguriert ist. In der Regel ist für die Erfassung von Änderungsdaten der Replikat-Set-Modus erforderlich. Als Nächstes erstellen Sie eine DMS-Replikationsinstanz und definieren Quell- und Zielendpunkte mit den erforderlichen Verbindungsdetails. Nach der Validierung der Endpunkte konfigurieren und starten Sie eine Migrationsaufgabe, die das vollständige Laden der Daten, die fortlaufende Replikation oder beides umfassen kann.

Erstellen Sie einen neuen Zielcluster

Siehe Neuen Zielcluster erstellen.

Quelle konfigurieren

MongoDB und Amazon DocumentDB können je nach Szenario beide als Migrationsquellen dienen:

- MongoDB als Quelle — Häufig bei der Migration von einer lokalen oder einer selbstverwalteten MongoDB zu einer Amazon DocumentDB oder anderen Datenbankdiensten. AWS Erfordert die

Ausführung im Replikatsatzmodus mit einem ausreichend großen Oplog (stellen Sie sicher, dass es so dimensioniert ist, dass alle Operationen während der Volllast ausgeführt werden können), um die Erfassung von Änderungsdaten während der Migration zu unterstützen.

- Amazon DocumentDB als Quelle — Wird in der Regel für regionsübergreifende Replikation, Versions-Upgrades oder die Migration zu anderen Datenbankdiensten wie MongoDB Atlas verwendet. Erfordert die Erfassung laufender Änderungen `change_stream_log_retention_duration` während der Migration, Change Streams aktivieren indem der Parameter in der Cluster-Parametergruppe gesetzt wird. Stellen Sie sicher, dass Ihre `change_stream_log_retention_duration` Einstellung groß genug ist, um die Zeit abzudecken, die für den vollständigen Ladevorgang benötigt wird.

Bevor Sie mit der Migration beginnen, konfigurieren Sie Ihre Quelle so, dass der AWS DMS Zugriff möglich ist.

Erstellen Sie einen MongoDB-Benutzer mit den richtigen Berechtigungen:

```
db.createUser({
  user: "dmsUser",
  pwd: "<password>",
  roles: [{ role: "readAnyDatabase", db: "admin" }]
})
```

Konfigurieren Sie das Netzwerk und die Authentifizierung.

Bei der Konfiguration der Netzwerkkonnektivität für die Migration von MongoDB zu DMS:

EC2-hosted MongoDB-Quelle

- Ändern Sie die EC2-Sicherheitsgruppe, um eingehenden Datenverkehr von der Sicherheitsgruppe der DMS-Replikationsinstanz zuzulassen.
- Fügen Sie eine Regel für den TCP-Port 27017 (oder Ihren benutzerdefinierten MongoDB-Port) hinzu.
- Verwenden Sie die Sicherheitsgruppen-ID der DMS-Replikationsinstanz als Quelle für eine präzise Zugriffskontrolle.
- Stellen Sie sicher, dass das Subnetz der EC2-Instance eine Route zum Subnetz der DMS-Replikationsinstanz hat.

On-premises MongoDB-Quelle

- Konfigurieren Sie Ihre Firewall so, dass eingehende Verbindungen von den öffentlichen IP-Adressen der DMS-Replikationsinstanz zugelassen werden.
- Wenn Sie ein VPN verwenden Direct Connect , stellen Sie sicher, dass das Routing zwischen Ihrem Netzwerk und der VPC, die die DMS-Instanz enthält, korrekt ist.
- Testen Sie die Konnektivität mit Telnet- oder NC-Befehlen vom DMS-Subnetz zu Ihrem MongoDB-Server.

MongoDB Atlas-Quelle

- Fügen Sie der MongoDB Atlas-IP-Zulassungsliste die IP-Adressen einer DMS-Replikationsinstanz hinzu.
- Konfigurieren Sie das VPC-Peering zwischen AWS VPC und MongoDB Atlas VPC, wenn Atlas ausgeführt wird. AWS
- AWS PrivateLink Richten Sie private Konnektivität (Enterprise-Stufe) ein, wenn Sie auf einem anderen Cloud-Anbieter ausgeführt werden.
- Erstellen Sie einen dedizierten Benutzer mit den entsprechenden read/write Berechtigungen.
- Verwenden Sie eine MongoDB Atlas-Verbindungszeichenfolge, bei der der SSL-Modus auf „verify-full“ gesetzt ist.
- Stellen Sie sicher, dass die Log-Größe für die Dauer der Migration ausreicht.

Amazon DocumentDB-Quelle

Konfigurieren Sie Ihre Amazon DocumentDB-Quellsicherheitsgruppe so, dass eingehender Datenverkehr von der Sicherheitsgruppe der DMS-Replikationsinstanz zugelassen wird.

Erstellen Sie eine DMS-Replikationsinstanz

Verwenden Sie DMS Buddy on GitHub , um eine optimale Migrationsinfrastruktur mit optimalen DMS-Einstellungen und Instanzgrößen zu erstellen. Wenn Sie es vorziehen, manuell zu konfigurieren, gehen Sie wie folgt vor:

1. Wählen Sie in der AWS DMS Konsole im Navigationsbereich Migrieren oder Replizieren und dann Bereitgestellte Instanzen aus.
2. Geben Sie Details zur Replikationsinstanz ein:
   - Instanzname: Wählen Sie einen eindeutigen Namen.

- Instanzklasse: Wählen Sie basierend auf der Arbeitslast aus. Beispiel: dms.r7i.large (kleine Workloads), dms.r7i.4xlarge (große Workloads).
- Version der Engine: 3.5.4
- Zugewiesener Speicher: Die Standardeinstellung ist 50 GB (bei Bedarf erhöhen). Dies wird durch die Dokumentgröße updates/second und die Dauer des vollständigen Ladevorgangs bestimmt.
- Multi-AZ Bereitstellung: Aktivieren Sie diese Option, falls erforderlich, für hohe Verfügbarkeit.
- Wählen Sie dieselbe VPC wie Amazon DocumentDB.
- Stellen Sie sicher, dass Sicherheitsgruppen eingehenden Datenverkehr von der Quelle und Amazon DocumentDB zulassen.

3. Wählen Sie Create replication instance und warten Sie, bis der Status verfügbar ist.

DMS-Endpunkte erstellen

Erstellen Sie einen Quellendpunkt

Für eine MongoDB-Quelle

1. Wählen Sie in der DMS-Konsole im Navigationsbereich Migrieren oder replizieren und dann Endpoints aus.
2. Wählen Sie Endpunkt erstellen aus.
3. Wählen Sie auf der Seite Endpunkt erstellen die Option Quellendpunkt aus.
4. Gehen Sie im Abschnitt Endpunktkonfiguration wie folgt vor:

   - Geben Sie eine eindeutige und aussagekräftige Endpunkt-ID ein (z. B. „mongodb-source“).
   - Wählen Sie MongoDB als Quell-Engine.
   - Wählen Sie für Zugriff auf Endpunktdatenbank die Option Zugriffsinformationen manuell bereitstellen aus.
   - Geben Sie als Servernamen Ihren *MongoDB server DNS name/IP address* ein.
   - Geben Sie für Port 27017 ein (Standard-MongoDB-Port).
   - Wählen Sie für den Authentifizierungsmodus den entsprechenden Modus für Ihre Anwendung (password/SSL) (Standard ist Secrets Manager).
   - Wenn der Authentifizierungsmodus Passwort ist, geben Sie Folgendes an:
     - Benutzername und Passwort: Geben Sie die MongoDB-Anmeldeinformationen ein.

- Datenbankname: Ihr Quelldatenbankname.
- Authentifizierungsmechanismus: SCRAM-SHA-1 (Standard) oder geeigneter Mechanismus

5. Belassen Sie für den Metadatenmodus die Standardeinstellung Dokument.
6. Zusätzliche Verbindungsattribute:

- authSource=admin (wenn die Authentifizierungsdatenbank unterschiedlich ist)
- <your-replica-set-name>ReplicaSet= (erforderlich für CDC)

Für eine Amazon DocumentDB-Quelle

1. Wählen Sie in der DMS-Konsole im Navigationsbereich Migrieren oder Replizieren und dann Endpoints aus.
2. Wählen Sie Endpunkt erstellen aus.
3. Wählen Sie auf der Seite Endpunkt erstellen die Option Quellendpunkt aus.
4. Gehen Sie im Abschnitt Endpunktkonfiguration wie folgt vor:

- Geben Sie eine eindeutige und aussagekräftige Endpunkt-ID ein (z. B. „docdb-source“).
- Wählen Sie Amazon DocumentDB als Quell-Engine.
- Wählen Sie für Zugriff auf Endpunktdatenbank die Option Zugriffsinformationen manuell bereitstellen aus.
- Geben Sie als Servernamen Ihren *source Amazon DocumentDB cluster endpoint* ein.
- Geben Sie 27017 als Port ein (Standardport für Amazon DocumentDB).
- Wählen Sie für den SSL-Modus verify-full (empfohlen für Amazon DocumentDB).
- Wählen Sie für CA-Zertifikat das Amazon RDS-Root-CA-Zertifikat aus.
- Wählen Sie für den Authentifizierungsmodus den für Ihre Anwendung geeigneten Modus (password/SSL) (die Standardeinstellung ist Secrets Manager).
- Wenn der Authentifizierungsmodus Passwort ist, geben Sie Folgendes an:
  - Benutzername und Passwort: Geben Sie die Amazon DocumentDB-Anmeldeinformationen ein.
  - Datenbankname: Ihr Quelldatenbankname.
  - Authentifizierungsmechanismus: SCRAM-SHA-1 (Standard) oder geeigneter Mechanismus

5. Belassen Sie für den Metadatenmodus die Standardeinstellung Dokument.

Erstellen Sie einen Zielendpunkt (Amazon DocumentDB)

1. Wählen Sie in der DMS-Konsole im Navigationsbereich Migrieren oder Replizieren und dann Endpoints aus.
2. Wählen Sie Endpunkt erstellen aus.
3. Wählen Sie auf der Seite Endpunkt erstellen die Option Zielendpunkt aus.
4. Gehen Sie im Abschnitt Endpunktkonfiguration wie folgt vor:
   - Geben Sie eine eindeutige und aussagekräftige Endpunkt-ID ein (z. B. „docdb-target“).
   - Wählen Sie Amazon DocumentDB als Ziel-Engine.
   - Wählen Sie für Zugriff auf die Endpunktdatenbank die Methode aus, mit der Sie den Zugriff auf die Datenbank authentifizieren möchten:
     - Wenn Sie AWS Secrets Manager wählen, wählen Sie im Feld Secret das Geheimnis aus, in dem Sie Ihre Amazon DocumentDB-Anmeldeinformationen speichern.
     - Wenn Sie „Zugangsinformationen manuell bereitstellen“ wählen:
       - Geben Sie als Servername Ihren ein*target Amazon DocumentDB cluster endpoint*.
       - Geben Sie 27017 als Port ein (Standardport für Amazon DocumentDB).
       - Wählen Sie für den SSL-Modus verify-full (empfohlen für Amazon DocumentDB).
       - Laden Sie für das CA-Zertifikat das CA-Zertifikatspaket für die SSL-Überprüfung herunter und geben Sie es an.
       - Wählen Sie für den Authentifizierungsmodus den für Ihre Anwendung geeigneten Modus (password/SSL) (die Standardeinstellung ist Secrets Manager).
       - Wenn der Authentifizierungsmodus Passwort ist, geben Sie Folgendes an:
         - Benutzername und Passwort: Geben Sie die Amazon DocumentDB-Anmeldeinformationen ein.
         - Datenbankname: Ihr Quelldatenbankname.
         - Authentifizierungsmechanismus: SCRAM-SHA-1 (Standard) oder geeigneter Mechanismus
5. Belassen Sie für den Metadatenmodus die Standardeinstellung Dokument.

Replikationsaufgabe erstellen

1. Wählen Sie in der DMS-Konsole im Navigationsbereich Migrieren oder replizieren und dann Aufgaben aus.
2. Wählen Sie Create task aus.
3. Gehen Sie auf der Seite „Aufgabe erstellen“ im Abschnitt Aufgabenkonfiguration wie folgt vor:
   - Geben Sie eine eindeutige und aussagekräftige Task-ID ein (z. B. „mongodb-docdb-replication“).
   - Wählen Sie im Dropdownmenü Quelldatenbank-Endpunkt den Quellendpunkt aus, den Sie zuvor erstellt haben.
   - Wählen Sie im Dropdownmenü Zieldatenbankendpunkt den Zielendpunkt aus, den Sie zuvor erstellt haben.
   - Wählen Sie im Dropdownmenü Bereitgestellte Instanz die zuvor erstellte Replikationsinstanz aus.
   - Wählen Sie als Aufgabentyp die Option Migrieren und replizieren aus.
   - Wie lange beabsichtigen Sie, die Replikation von der Quelle zum Ziel fortzusetzen? , wählen Sie Für eine begrenzte Zeit.
4. Im Abschnitt Einstellungen:
   - Wählen Sie für den Vorbereitungsmodus Zieltabelle die Option Nichts tun aus. Andernfalls werden die Sammlungen und Indizes, die bereits erstellt wurden, gelöscht.
   - Markieren Sie für Aufgabenprotokolle das Kontrollkästchen CloudWatch Protokolle einschalten.
   - Behalten Sie die Standardwerte für alle anderen Einstellungen bei.
5. Zurück oben im Abschnitt Einstellungen wählen Sie für den Bearbeitungsmodus den JSON-Editor und legen Sie die folgenden Attribute fest:

```
{
  "TargetMetadata": {
    "ParallelApplyThreads": 5
  },
  "FullLoadSettings": {
    "MaxFullLoadSubTasks": 16
  }
}
```

Diese Einstellung `MaxFullLoadSubTasks` basiert auf der Anzahl Ihrer Sammlungen und der Instanzkapazität. Verwenden Sie niedrigere Werte (2—4) für weniger oder kleinere Sammlungen und höhere Werte (8—16) für viele große Sammlungen. Informationen zur automatischen Größenanpassung finden Sie unter DMS Buddy auf. GitHub

6. Fügen Sie im Abschnitt Tabellenzuordnungen eine neue Auswahlregel hinzu:

   - Fügen Sie für den Schemanamen die zu migrierende Quelldatenbank hinzu. Verwenden Sie %, um mehrere Datenbanken anzugeben.
   - Fügen Sie für den Namen der Schematabelle die zu migrierende Quellsammlung hinzu. Verwenden Sie%, um mehrere Sammlungen anzugeben.
   - Belassen Sie für Aktion die Standardeinstellung Include

7. Fügen Sie für große Sammlungen (über 100 GB) die Regel für Tabelleneinstellungen hinzu:

   - Fügen Sie für den Schemanamen die zu migrierende Quelldatenbank hinzu. Verwenden Sie %, um mehrere Datenbanken anzugeben.
   - Fügen Sie für den Namen der Schematabelle die zu migrierende Quellsammlung hinzu. Verwenden Sie%, um mehrere Sammlungen anzugeben.
   - Legen Sie die Anzahl der Partitionen auf der Grundlage Ihrer Arbeitslast fest. Stellen Sie für eine kleine Anzahl großer Sammlungen die Partitionen ≥ ein, `MaxFullLoadSubTasks` um alle Mitarbeiter zu beschäftigen. Für viele Sammlungen, die parallel geladen werden, setzen Sie Partitionen auf ≤, `MaxFullLoadSubTasks` da die Anzahl der Sammlungen selbst für eine ausreichende Parallelität sorgt.

8. Vergewissern Sie sich, dass im Abschnitt zur Bewertung vor der Migration die Option deaktiviert ist. Die Bewertung vor der Migration wird für Amazon DocumentDB als Ziel nicht unterstützt und muss deaktiviert werden.

Offline-Migration

In diesem Abschnitt wird der Prozess zur Durchführung einer Offline-Migration von einer selbstverwalteten MongoDB-Instance zu Amazon DocumentDB mithilfe nativer MongoDB-Tools beschrieben: und. `mongodump` `mongorestore`

Voraussetzungen

Quelle: MongoDB-Anforderungen

- Zugriff auf die MongoDB-Quellinstanz mit entsprechenden Berechtigungen.
- Installieren`mongodump`. falls erforderlich (es wird während einer MongoDB-Installation installiert).
- Stellen Sie sicher, dass genügend Speicherplatz für die Dump-Dateien vorhanden ist.

Target, Amazon DocumentDB-Anforderungen

- Stellen Sie sicher, dass Sie einen Amazon DocumentDB-Cluster bereitgestellt haben.
- Stellen Sie sicher, dass sich in derselben VPC wie Amazon DocumentDB eine EC2-Instance befindet, um die Migration zu erleichtern.
- Die Netzwerkkonnektivität zwischen Ihrer Quellumgebung und Amazon DocumentDB muss verfügbar sein.
- mongorestore muss auf der EC2-Migrationsinstanz installiert sein.
- Für den Zugriff auf Amazon DocumentDB müssen entsprechende IAM-Berechtigungen konfiguriert werden.

Allgemeine Anforderungen

- AWS CLI muss konfiguriert werden (wenn AWS Dienste für den Zwischenspeicher verwendet werden)
- Für die Datenübertragung muss ausreichend Bandbreite verfügbar sein.
- Das Ausfallzeitfenster sollte genehmigt werden (bei einer Live-Migration sollten Sie andere Ansätze in Betracht ziehen)

Bereiten Sie einen Amazon DocumentDB-Cluster vor

Siehe Neuen Zielcluster erstellen.

Führen Sie den Datendump (mongodump) durch

Wählen Sie eine der folgenden Optionen, um eine Dump-Datei zu erstellen:

- Option 1: Einfach

```
mongodump \
  --uri="mongodb://<user>:<password>@<source_endpoint>:27017/<database>" \
  --out=/path/to/dump
```

- Option 2: Bessere Steuerung und Leistung

  Komprimiert die Ausgabe mit gzip, gibt Sammlungen parallel aus, aktiviert TLS und liest aus einem sekundären Replikat, falls verfügbar:

```
mongodump \
  --uri="mongodb://<user>:<password>@<source_endpoint>:27017" \
  --out=/path/to/dump \
  --gzip \
  --numParallelCollections=4 \
  --tls \
  --authenticationDatabase=admin \
  --readPreference=secondaryPreferred
```

- Option 3: Große Datenbanken

  Speichert eine bestimmte Datenbank und Sammlung, filtert Dokumente nach Abfragen und gibt sie in ein einziges komprimiertes Archiv aus:

```
mongodump \
  --host=<source_endpoint> \
  --port=27017 \
  --username=<user> \
  --password=<password> \
  --db=<specific_db> \
  --collection=<specific_collection> \
  --query='{ "date": { "$gt": "2020-01-01" } }' \
  --archive=/path/to/archive.gz \
  --gzip \
  --tls
```

Überträgt Dumpdateien in die Wiederherstellungsumgebung

Wählen Sie basierend auf Ihrer Dumpgröße eine geeignete Methode:

- Klein — Kopieren Sie direkt auf Ihren Migrationscomputer (EC2-Instance, die Sie zuvor erstellt haben):

```
scp -r /path/to/dump user@migration-machine:/path/to/restore
```

- Mittel — Amazon S3 als Zwischenspeicher verwenden:

```
aws s3 cp --recursive /path/to/dump s3://your-bucket/mongodb-dump/
```

- Groß — Ziehen Sie bei sehr großen Datenbanken eine physische Übertragung in Betracht AWS DataSync .

Daten in Amazon DocumentDB wiederherstellen (mongorestore)

Bevor Sie den Wiederherstellungsvorgang starten, erstellen Sie die Indizes in Amazon DocumentDB. Sie können das Amazon DocumentDB Index-Tool verwenden, um Indizes zu exportieren und zu importieren.

Wählen Sie eine der folgenden Optionen, um Daten wiederherzustellen:

- Option 1: Grundlegende Wiederherstellung

```
mongorestore \
  --uri="mongodb://<user>:<password>@<target_endpoint>:27017" \
  --tls \
  --tlsCAFile=global-bundle.pem \
  /path/to/dump
```

- Option 2: Bessere Steuerung und Leistung

  Stellt mit Gzip komprimierte Dumps mit paralleler Erfassung und Einfügen von Dokumenten wieder her. Überspringt die Indexwiederherstellung, da Indizes vorab erstellt werden:

```
mongorestore \
  --uri="mongodb://<user>:<password>@<target_endpoint>:27017" \
  --tls \
  --tlsCAFile=global-bundle.pem \
  --gzip \
  --numParallelCollections=4 \
  --numInsertionWorkersPerCollection=4 \
  --noIndexRestore \
  /path/to/dump
```

- Option 3: Große Datenbanken oder bestimmte Steuerelemente

  Wiederherstellungen aus einem komprimierten Archiv mit Namespace-Filterung, um bestimmte Datenbanken und Sammlungen ein- oder auszuschließen. Überspringt die Indexwiederherstellung, da Indizes vorab erstellt werden:

```
mongorestore \
  --host=<target_endpoint> \
  --port=27017 \
  --username=<user> \
  --password=<password> \
  --tls \
  --tlsCAFile=global-bundle.pem \
  --archive=/path/to/archive.gz \
  --gzip \
  --nsInclude="db1.*" \
  --nsExclude="db1.sensitive_data" \
  --noIndexRestore
```

Überwachen

Dieser Abschnitt enthält einen detaillierten Überwachungsprozess, mit dem Sie den Fortschritt, die Leistung und den Zustand einer laufenden Migration verfolgen können.

Die Überwachungsschritte gelten unabhängig von der Migrationsmethode (AWS DMS mongodump/mongorestore, oder anderen Tools).

AWS DMS Migrationsüberwachung (falls zutreffend)

Überwachen Sie die folgenden wichtigen CloudWatch Kennzahlen:

Metriken für die Volllastphase

- FullLoadThroughputBandwidthTarget— Netzwerkbandbreite (KB/second) bei Volllast
- FullLoadThroughputRowsTarget— Anzahl der pro Sekunde rows/documents geladenen Dateien
- FullLoadThroughputTablesTarget— Anzahl der tables/collections abgeschlossenen Dateien pro Minute
- FullLoadProgressPercent— Prozentsatz der abgeschlossenen Volllast
- TablesLoaded— Anzahl der tables/collections erfolgreich geladenen
- TablesLoading— Anzahl der tables/collections aktuell geladenen Dateien
- TablesQueued— Anzahl der Personen, die tables/collections darauf warten, geladen zu werden
- TablesErrored— Anzahl der Dateien tables/collections , die nicht geladen werden konnten

Metriken für die CDC-Phase

- CDCLatencyTarget— Zeitverzögerung (Sekunden) zwischen Quellwechsel und Zielanwendung
- CDCLatencySource— Zeitverzögerung (Sekunden) zwischen der Änderung der Quelle und dem Lesen durch das DMS
- CDCThroughputRowsTarget— Während der laufenden Replikation werden Zeilen pro Sekunde angewendet
- CDCThroughputBandwidthTarget— Netzwerkbandbreite (KB/second) während CDC
- CDCIncomingChanges— Anzahl der von der Quelle empfangenen Änderungsereignisse
- CDCChangesMemoryTarget— Verwendeter Speicher (MB) zum Speichern von Änderungen auf der Zielseite

Metriken für Ressourcen

- CPUUtilization — CPU-Auslastung der Replikationsinstanz
- FreeableMemory— Verfügbarer Speicher auf der Replikationsinstanz
- FreeStorageSpace— Verfügbarer Speicher auf der Replikationsinstanz
- NetworkTransmitThroughput— Netzwerkdurchsatz für die Replikationsinstanz
- NetworkReceiveThroughput— Netzwerkdurchsatz für die Replikationsinstanz

Fehler-Metriken

- ErrorsCount— Gesamtzahl der Fehler während der Migration
- TableErrorsCount— Anzahl der tabellenspezifischen Fehler
- RecordsErrorsCount— Anzahl der datensatzspezifischen Fehler

Erstellen Sie CloudWatch Alarme für kritische Kennzahlen, z. B. `CDCLatencyTarget` um Benachrichtigungen `CPUUtilization` zu erhalten, wenn sich die Migrationsleistung verschlechtert.

DMS-Protokolle (Protokolle) CloudWatch

1. Gehen Sie zur Amazon CloudWatch Logs-Konsole.
2. Suchen Sie nach Ihrer Log-Gruppe und wählen Sie sie aus. Es wird ähnlich aussehen wie „dms-tasks —“.

3. Suchen Sie nach Log-Streams, die möglicherweise Fehlerinformationen enthalten:
   - Streams mit „Fehler“ im Namen
   - Streams mit Aufgaben-IDs oder Endpunktnamen
   - Die neuesten Log-Streams zum Zeitpunkt Ihrer Migration
4. Suchen Sie in diesen Streams nach Schlüsselwörtern wie:
   - „Fehler“
   - „Ausnahme“
   - „gescheitert“
   - „Warnung“

DMS-Aufgabenstatus (verwenden AWS CLI)

```
aws dms describe-replication-tasks \
  --filters Name=replication-task-id,Values=<task_id> \
  --query "ReplicationTasks[0].Status"
```

Erwarteter Statusablauf:

erstellen → bereit → läuft → stoppen → gestoppt (oder fehlgeschlagen)

Monitor mit **`Docdb-Dashboarder`**

Das `docdb-dashboarder` Tool bietet eine umfassende Überwachung für Amazon DocumentDB-Cluster, indem es automatisch Dashboards mit wichtigen Leistungskennzahlen generiert CloudWatch . Diese Dashboards zeigen wichtige Metriken auf Cluster-Ebene (Replikatverzögerung, Betriebszähler), Metriken auf Instance-Ebene (CPU, Arbeitsspeicher, Verbindungen) und Speichermetriken (Volumennutzung, Backup-Speicher) an. Für Migrationsszenarien bietet das Tool spezielle Dashboards, die den Migrationsfortschritt anhand von Metriken wie CDC-Replikationsverzögerung und Betriebsraten verfolgen. Die Dashboards können mehrere Cluster gleichzeitig überwachen und bieten Unterstützung für Instances. NVMe-backed Durch die Visualisierung dieser Kennzahlen können Teams proaktiv Leistungsengpässe erkennen, die Ressourcenzuweisung optimieren und den reibungslosen Betrieb ihrer Amazon DocumentDB-Bereitstellungen sicherstellen. Das Tool macht die manuelle Erstellung von Dashboards überflüssig und bietet gleichzeitig eine konsistente Überwachung in allen Umgebungen. Anweisungen zur

Einrichtung und erweiterte Konfigurationsoptionen finden Sie im Amazon DocumentDB Dashboarder Tool-Repository. GitHub

Validierung

Topics

- Checkliste für die Validierung
- Schema- und Indexvalidierung
- Datenstichprobenentnahme und Validierung auf Feldebene
- Validierung mit dem DataDiffer Tool

Dieser Abschnitt enthält einen detaillierten Validierungsprozess, um die Datenkonsistenz, Integrität und Anwendungskompatibilität nach der Migration sicherzustellen.

Die Validierungsschritte gelten unabhängig von der Migrationsmethode (AWS DMS mongodump/mongorestore, oder anderen Tools).

Checkliste für die Validierung

Stellen Sie sicher, dass die Anzahl der Dokumente in jeder Sammlung zwischen Quelle und Ziel übereinstimmt:

MongoDB-Quelle

```
mongosh \
  --host <source_endpoint> \
  --port 27017 \
  --username <user> \
  --password <password> \
  --eval "db.<collection>.countDocuments()"
```

Amazon DocumentDB-Ziel

```
mongosh \
  --host <target_endpoint> \
  --port 27017 \
  --username <user> \
  --password <password> \
  --tls \
```

```
  --tlsCAFile global-bundle.pem \
  --eval "db.<collection>.countDocuments()"
```

Schema- und Indexvalidierung

Stellen Sie Folgendes sicher:

- Alle Sammlungen sind im Ziel vorhanden.
- Indizes werden korrekt repliziert.
- Schemadefinitionen (falls erzwungen) sind identisch.

Überprüfen Sie die Sammlungen (Quelle oder Ziel)

```
mongosh --host <source_endpoint> \
  --port 27017 \
  --username <username> \
  --password <password> \
  --eval "db = db.getSiblingDB('<database>'); db.getCollectionNames();"

mongosh --host <target_endpoint> \
  --port 27017 \
  --username <username> \
  --password <password> \
  --tls --tlsCAFile global-bundle.pem \
  --eval "db = db.getSiblingDB('<database>'); db.getCollectionNames();"
```

Indizes überprüfen (Quelle oder Ziel)

```
mongosh --host <source_endpoint> \
  --port 27017 \
  --username <username> \
  --password <password> \
  --eval "db.<collection>.getIndexes()"

mongosh --host <target_endpoint> \
  --port 27017 \
  --username <username> \
  --password <password> \
  --tls --tlsCAFile global-bundle.pem \
  --eval "db.<collection>.getIndexes()"
```

Vergleichen Sie die Liste der Sammlungen, um sicherzustellen, dass keine Sammlungen fehlen oder zusätzliche Sammlungen vorhanden sind.

Überprüfen Sie Indizes, indem Sie Indexnamen, Schlüsseldefinitionen, eindeutige Einschränkungen und TTL-Indizes (falls vorhanden) überprüfen.

Überprüfen Sie die Schemavalidierungsregeln (wenn Sie die Schemavalidierung in MongoDB verwenden)

```
mongosh --host <source_endpoint> \
  --port 27017 \
  --username <username> \
  --password <password> \
  --eval "db.getCollectionInfos({name: '<collection>'})[0].options.validator"

mongosh --host <target_endpoint> \
  --port 27017 \
  --username <username> \
  --password <password> \
  --tls --tlsCAFile global-bundle.pem \
  --eval "db.getCollectionInfos({name: '<collection>'})[0].options.validator"
```

Datenstichprobenentnahme und Validierung auf Feldebene

Sie können Dokumente nach dem Zufallsprinzip auswählen und Felder zwischen Quelle und Ziel vergleichen.

Manuelle Probenentnahme

Rufe fünf zufällige Dokumente ab (Quelle):

```
mongosh --host <source_endpoint> \
  --port 27017 \
  --username <username> \
  --password <password> \
  --eval "db.<collection>.aggregate([{ \$sample: { size: 5 } }])"
```

Ruft dieselben Dokument-IDs ab (Ziel):

```
mongosh --host <target_endpoint> \
  --port 27017 \
  --username <username> \
```

```
  --password <password> \
  --tls --tlsCAFile global-bundle.pem \
  --eval "db.<collection>.find({ _id: { \$in: [<list_of_ids>] } })"
```

Automatische Probenentnahme

```
import pymongo

# Connect to source and target
source_client = pymongo.MongoClient("<source_uri>")
target_client = pymongo.MongoClient("<target_uri>",
                                    tls=True,
                                    tlsCAFile='global-bundle.pem')
source_db = source_client["<db_name>"]
target_db = target_client["<db_name>"]

# Compare 100 random documents
for doc in source_db["<collection>"].aggregate([{"$sample": {"size": 100}}]):
    target_doc = target_db["<collection>"].find_one({"_id": doc["_id"]})
    if target_doc != doc:
        print(f"Mismatch in _id: {doc['_id']}")
    else:
        print(f"Match: {doc['_id']}")
```

Validierung mit dem DataDiffer Tool

Das DataDiffer Tool on GitHub bietet eine zuverlässige Möglichkeit, Daten zwischen Quell- und Zieldatenbanken zu vergleichen.

Voraussetzungen

Die folgenden Voraussetzungen müssen vor der Installation des DataDiffer Tools erfüllt sein:

- Python3,7+
- PyMongo Bibliothek
- Netzwerkkonnektivität sowohl zu MongoDB-Quellclustern als auch zu Amazon DocumentDB-Zielclustern

Einrichtung und Installation

Klonen Sie das Repository und navigieren Sie zum Verzeichnis DataDiffer

```
git clone https://github.com/awslabs/amazon-documentdb-tools.git
cd amazon-documentdb-tools/migration/data-differ
```

Installieren Sie die erforderlichen Abhängigkeiten

```
pip install -r requirements.txt
```

Datenvalidierung ausführen

Erstellen Sie eine Konfigurationsdatei (z. B. config.json) mit Verbindungsdetails

```
{
  "source": {
    "uri": "mongodb://<username>:<password>@<source_endpoint>:27017/?replicaSet=rs0",
    "db": "your_database",
    "collection": "your_collection"
  },
  "target": {
    "uri": "mongodb://<username>:<password>@<target_endpoint>:27017/?
tls=true&tlsCAFile=global-bundle.pem&replicaSet=rs0&retryWrites=false",
    "db": "your_database",
    "collection": "your_collection"
  },
  "options": {
    "batch_size": 1000,
    "threads": 4,
    "sample_size": 0,
    "verbose": true
  }
}
```

Führen Sie das Tool aus DataDiffer

```
python differ.py --config config.json
```

Verwenden Sie bei großen Sammlungen Stichproben, um eine Teilmenge von Daten zu validieren

```
python differ.py --config config.json --sample-size 10000
```

Um mehrere Sammlungen zu validieren, erstellen Sie separate Konfigurationsdateien oder verwenden Sie den Batch-Modus

```
python differ.py --batch-config batch_config.json
```

Ergebnisse interpretieren

Das Tool gibt Folgendes aus:

- Gesamtzahl der Dokumente in Quelle und Ziel
- Anzahl übereinstimmender Dokumente
- Anzahl der fehlenden Dokumente
- Anzahl der Dokumente mit Unterschieden
- Detaillierter Bericht über Unterschiede (falls vorhanden)

Bewährte Methoden

Im Folgenden finden Sie bewährte Methoden bei der Verwendung des DataDiffer Tools:

- Schrittweise ausführen — Überprüfen Sie zunächst die Anzahl der Dokumente, nehmen Sie dann Stichproben wichtiger Dokumente vor und führen Sie schließlich bei Bedarf einen vollständigen Vergleich durch.
- Suchen Sie nach Schemaunterschieden — Amazon DocumentDB hat im Vergleich zu MongoDB einige Einschränkungen. Das Tool hebt inkompatible Datentypen oder Strukturen hervor.
- In Ruhephasen validieren — Führen Sie die Validierung aus, wenn nur wenige Schreibvorgänge erforderlich sind, um die Konsistenz zu gewährleisten.
- Überwachen Sie die Ressourcennutzung — Der Vergleichsprozess kann ressourcenintensiv sein. Passen Sie die Chargengröße und die Threadanzahl entsprechend an.
- Indizes validieren — Stellen Sie nach der Datenvalidierung sicher, dass alle erforderlichen Indizes auf dem Amazon DocumentDB-Zielcluster erstellt wurden.
- Ergebnisse der Dokumentenvalidierung — Führen Sie als Teil Ihrer Migrationsdokumentation Aufzeichnungen über die Validierungsergebnisse für jede Sammlung.

Migration vom Couchbase Server

Topics

- Einführung

- Vergleich mit Amazon DocumentDB
- Erkennung
- Planung
- Migration
- Validierung

Einführung

In diesem Handbuch werden die wichtigsten Punkte vorgestellt, die bei der Migration von Couchbase Server zu Amazon DocumentDB zu beachten sind. Es werden Überlegungen für die Ermittlungs-, Planungs-, Durchführungs- und Validierungsphasen Ihrer Migration erläutert. Außerdem wird erklärt, wie Offline- und Online-Migrationen durchgeführt werden.

Vergleich mit Amazon DocumentDB

	Couchbase Server	Amazon DocumentDB
Organisation der Daten	In den Versionen 7.0 und höher sind Daten in Buckets, Bereichen und Sammlungen organisiert. In früheren Versionen sind Daten in Buckets organisiert.	Die Daten sind in Datenbanken und Sammlungen organisiert.
Kompatibilität	Für jeden Dienst gibt es separate APIs (z. B. Daten, Index, Suche usw.). Sekundäre Suchvorgänge verwenden SQL++ (früher bekannt als N1QL); eine Abfragesprache, die auf ANSI-standard SQL basiert und daher vielen Entwicklern vertraut ist.	Amazon DocumentDB ist mit der MongoDB-API kompatibel.
Architektur	Der Speicher ist an jede Cluster-I nstance angehängt. Sie können die Rechenleistung nicht unabhängig vom Speicher skalieren.	Amazon DocumentDB wurde für die Cloud und zur Vermeidung der Einschränkungen herkömmli cher Datenbankarchitekturen

	Couchbase Server	Amazon DocumentDB
		entwickelt. Die Rechen- und Speicherschicht sind in Amazon DocumentDB getrennt, und die Rechenschicht kann unabhängig vom Speicher skaliert werden.
Fügen Sie Lesekapazität bei Bedarf hinzu	Cluster können durch Hinzufügen von Instanzen skaliert werden. Da der Speicher an die Instanz angehängt ist, auf der der Service ausgeführt wird, hängt die Zeit, die für die Skalierung benötigt wird, von der Datenmenge ab, die auf die neue Instanz verschoben oder neu verteilt werden muss.	Sie können eine Leseskalierung für Ihren Amazon DocumentDB-Cluster erreichen, indem Sie bis zu 15 Amazon DocumentDB-Replikate im Cluster erstellen. Es gibt keine Auswirkungen auf die Speicherebene.
Schnelle Wiederherstellung nach einem Knotenausfall	Cluster verfügen über automatische Failover-Funktionen, aber die Zeit, bis der Cluster wieder seine volle Stärke erreicht, hängt von der Datenmenge ab, die auf die neue Instance verschoben werden muss.	Amazon DocumentDB kann in der Regel innerhalb von 30 Sekunden ein Failover für die primäre Komponente durchführen und den Cluster unabhängig von der Datenmenge im Cluster innerhalb von 8 bis 10 Minuten wieder auf volle Stärke zurücksetzen.
Skalieren Sie den Speicher, wenn die Datenmenge wächst	Bei selbstverwalteten Clustern werden Speicher und iOS nicht automatisch skaliert.	Amazon DocumentDB-Speicher und iOS werden automatisch skaliert.

	Couchbase Server	Amazon DocumentDB
Daten sichern, ohne die Leistung zu beeinträchtigen	Backups werden vom Backup-Dienst durchgeführt und sind standardmäßig nicht aktiviert. Da Speicher und Rechenleistung nicht getrennt sind, kann dies Auswirkungen auf die Leistung haben.	Amazon DocumentDB-Backups sind standardmäßig aktiviert und können nicht deaktiviert werden. Backups werden von der Speicherebene verarbeitet, sodass sie keine Auswirkungen auf die Rechenebene haben. Amazon DocumentDB unterstützt die Wiederherstellung aus einem Cluster-Snapshot und die Wiederherstellung auf einen bestimmten Zeitpunkt.
Haltbarkeit der Daten	In einem Cluster können maximal 3 Replikatkopien von Daten vorhanden sein, also insgesamt 4 Kopien. Auf jeder Instanz, auf der der Datendienst ausgeführt wird, sind 1, 2 oder 3 Replikatkopien der Daten aktiv.	Amazon DocumentDB verwaltet sechs Kopien der Daten, unabhängig davon, wie viele Recheninstanzen es gibt, mit einem Schreibquorum von 4 und Persist True. Kunden erhalten eine Bestätigung, nachdem auf der Speicherebene 4 Kopien der Daten gespeichert wurden.
Konsistenz	Sofortige K/V Betriebskonsistenz wird unterstützt. Das Couchbase SDK leitet K/V Anfragen an die spezifische Instanz weiter, die die aktive Kopie der Daten enthält. Sobald ein Update bestätigt wurde, kann der Client dieses Update garantiert lesen. Die Replikation von Updates auf andere Dienste (Index, Suche, Analytik, Eventing) erfolgt letztendlich konsistent.	Amazon DocumentDB-Replikate sind letztendlich konsistent. Wenn sofortige Konsistenzlesevorgänge erforderlich sind, kann der Client von der primären Instance lesen.

	Couchbase Server	Amazon DocumentDB
Replikation	Cross-Data Center Replication (XDCR) ermöglicht eine gefilterte, passive/active aktiv-aktive Replikation von Daten in vielen:vielen Topologien.	Die globalen Amazon DocumentDB-Cluster bieten eine aktiv-passive Replikation in 1:many (bis zu 10) Topologien.

Erkennung

Die Migration zu Amazon DocumentDB erfordert ein gründliches Verständnis der bestehenden Datenbank-Arbeitslast. Bei der Workload-Erkennung werden die Konfiguration und die Betriebsmerkmale Ihres Couchbase-Clusters — Datensatz, Indizes und Workload — analysiert, um einen nahtlosen Übergang mit minimalen Unterbrechungen zu gewährleisten.

Cluster-Konfiguration

Couchbase verwendet eine serviceorientierte Architektur, bei der jede Funktion einem Dienst entspricht. Führen Sie den folgenden Befehl für Ihren Couchbase-Cluster aus, um festzustellen, welche Dienste verwendet werden (siehe Informationen über Knoten abrufen):

```
curl -v -u <administrator>:<password> \
  http://<ip-address-or-hostname>:<port>/pools/nodes | \
  jq '[.nodes[].services[]] | unique'
```

Beispielausgabe:

```
[
  "backup",
  "cbas",
  "eventing",
  "fts",
  "index",
  "kv",
  "n1ql"
]
```

Die Couchbase-Dienste umfassen Folgendes:

Datenservice (kv)

Der Datendienst ermöglicht read/write den Zugriff auf Daten im Speicher und auf der Festplatte.

Amazon DocumentDB unterstützt K/V Operationen mit JSON-Daten über die MongoDB-API.

Abfragedienst (n1ql)

Der Abfragedienst unterstützt die Abfrage von JSON-Daten über SQL++.

Amazon DocumentDB unterstützt die Abfrage von JSON-Daten über die MongoDB-API.

Indexdienst (Index)

Der Indexdienst erstellt und verwaltet Indizes für Daten und ermöglicht so schnellere Abfragen.

Amazon DocumentDB unterstützt einen Standard-Primärindex und die Erstellung von Sekundärindizes für JSON-Daten über die MongoDB-API.

Suchdienst (fts)

Der Suchdienst unterstützt die Erstellung von Indizes für die Volltextsuche.

Mit der systemeigenen Volltextsuchfunktion von Amazon DocumentDB können Sie mithilfe der API einfache Textabfragen für große Textdatensätze mithilfe von Textindizes für spezielle Zwecke durchführen. MongoDB Für erweiterte Funktionen wie Fuzzy-Suche, mehrsprachige Suche und sammlungsübergreifende Suche verwenden Sie die Amazon DocumentDB Zero-ETL-Integration mit Amazon Service. OpenSearch

Analysedienst (cbas)

Der Analysedienst unterstützt die Analyse von JSON-Daten nahezu in Echtzeit.

Amazon DocumentDB unterstützt Ad-hoc-Abfragen zu JSON-Daten über die MongoDB-API. Sie können auch komplexe Abfragen für Ihre JSON-Daten in Amazon DocumentDB ausführen, indem Sie Apache Spark verwenden, der auf Amazon EMR läuft.

Veranstaltungsservice (Eventing)

Der Eventing-Service führt benutzerdefinierte Geschäftslogik als Reaktion auf Datenänderungen aus.

Amazon DocumentDB automatisiert ereignisgesteuerte Workloads, indem jedes Mal AWS Lambda Funktionen aufgerufen werden, wenn sich Daten in Ihrem Amazon DocumentDB-Cluster ändern.

Backup-Service (Sicherung)

Der Backup-Service plant vollständige und inkrementelle Datensicherungen und führt frühere Datensicherungen zusammen.

Amazon DocumentDB sichert Ihre Daten kontinuierlich auf Amazon S3 mit einer Aufbewahrungsfrist von 1—35 Tagen, sodass Sie sie innerhalb des Aufbewahrungszeitraums für Backups schnell wiederherstellen können. Amazon DocumentDB erstellt im Rahmen dieses kontinuierlichen Sicherungsprozesses auch automatische Schnappschüsse Ihrer Daten. Sie können die Sicherung und Wiederherstellung von Amazon DocumentDB auch mit verwalten. AWS Backup.

Betriebliche Eigenschaften

Verwenden Sie das Discovery Tool für Couchbase, um die folgenden Informationen über Ihren Datensatz, Ihre Indizes und Ihre Arbeitslast zu erhalten. Diese Informationen helfen Ihnen bei der Größe Ihres Amazon DocumentDB-Clusters.

Dataset

Das Tool ruft die folgenden Bucket-, Umfang- und Sammlungsinformationen ab:

1. Bucket-Name
2. Bucket-Typ
3. Name des Bereichs
4. Name der Sammlung
5. Gesamtgröße (Byte)
6. Gesamtzahl der Artikel
7. Artikelgröße (Byte)

Indizes

Das Tool ruft die folgenden Indexstatistiken und alle Indexdefinitionen für alle Buckets ab. Beachten Sie, dass Primärindizes ausgeschlossen sind, da Amazon DocumentDB automatisch einen Primärindex für jede Sammlung erstellt.

1. Bucket-Name
2. Name des Bereichs
3. Name der Sammlung

4. Name des Indexes
5. Indexgröße (Byte)

Workload

Das Tool ruft K/V N1QL-Abfrage-Metriken ab. K/V Metrikwerte werden auf Bucket-Ebene erfasst und SQL++-Metriken werden auf Cluster-Ebene erfasst.

Die Befehlszeilenoptionen des Tools lauten wie folgt:

```
python3 discovery.py \
  --username <source cluster username> \
  --password <source cluster password> \
  --data_node <data node IP address or DNS name> \
  --admin_port <administration http REST port> \
  --kv_zoom <get bucket statistics for specified interval> \
  --tools_path <full path to Couchbase tools> \
  --index_metrics <gather index definitions and SQL++ metrics> \
  --indexer_port <indexer service http REST port> \
  --n1ql_start <start time for sampling> \
  --n1ql_step <sample interval over the sample period>
```

Hier sehen Sie ein Beispiel für einen Befehl.

```
python3 discovery.py \
  --username username \
  --password ******** \
  --data_node "http://10.0.0.1" \
  --admin_port 8091 \
  --kv_zoom week \
  --tools_path "/opt/couchbase/bin" \
  --index_metrics true \
  --indexer_port 9102 \
  --n1ql_start -60000 \
  --n1ql_step 1000
```

K/V Die Metrikwerte basieren auf Stichproben, die in der letzten Woche alle 10 Minuten abgefragt wurden (siehe HTTP-Methode und URI). SQL++-Metrikwerte basieren auf Stichproben, die in den letzten 60 Sekunden alle 1 Sekunden abgefragt wurden (siehe Allgemeine Bezeichnungen). Die Ausgabe des Befehls erfolgt in den folgenden Dateien:

collection-stats.csv — Bucket-, Umfang- und Sammlungsinformationen

```
bucket,bucket_type,scope_name,collection_name,total_size,total_items,document_size
beer-sample,membase,_default,_default,2796956,7303,383
gamesim-sample,membase,_default,_default,114275,586,196
pillowfight,membase,_default,_default,1901907769,1000006,1902
travel-sample,membase,inventory,airport,547914,1968,279
travel-sample,membase,inventory,airline,117261,187,628
travel-sample,membase,inventory,route,13402503,24024,558
travel-sample,membase,inventory,landmark,3072746,4495,684
travel-sample,membase,inventory,hotel,4086989,917,4457
...
```

index-stats.csv — Indexnamen und -größen

```
bucket,scope,collection,index-name,index-size
beer-sample,_default,_default,beer_primary,468144
gamesim-sample,_default,_default,gamesim_primary,87081
travel-sample,inventory,airline,def_inventory_airline_primary,198290
travel-sample,inventory,airport,def_inventory_airport_airportname,513805
travel-sample,inventory,airport,def_inventory_airport_city,487289
travel-sample,inventory,airport,def_inventory_airport_faa,526343
travel-sample,inventory,airport,def_inventory_airport_primary,287475
travel-sample,inventory,hotel,def_inventory_hotel_city,497125
...
```

kv-stats.csv — Metriken für alle Buckets abrufen, festlegen und löschen

```
bucket,gets,sets,deletes
beer-sample,0,0,0
gamesim-sample,0,0,0
pillowfight,369,521,194
travel-sample,0,0,0
```

n1ql-stats.csv — SQL++ wählt Metriken für den Cluster aus, löscht sie und fügt sie ein

```
selects,deletes,inserts
0,132,87
```

indexes- <bucket-name>.txt — Indexdefinitionen aller Indizes im Bucket. Beachten Sie, dass Primärindizes ausgeschlossen sind, da Amazon DocumentDB automatisch einen Primärindex für jede Sammlung erstellt.

```
CREATE INDEX `def_airportname` ON `travel-sample`(`airportname`)
CREATE INDEX `def_city` ON `travel-sample`(`city`)
CREATE INDEX `def_faa` ON `travel-sample`(`faa`)
CREATE INDEX `def_icao` ON `travel-sample`(`icao`)
CREATE INDEX `def_inventory_airport_city` ON `travel-
sample`.`inventory`.`airport`(`city`)
CREATE INDEX `def_inventory_airport_faa` ON `travel-
sample`.`inventory`.`airport`(`faa`)
CREATE INDEX `def_inventory_hotel_city` ON `travel-sample`.`inventory`.`hotel`(`city`)
CREATE INDEX `def_inventory_landmark_city` ON `travel-
sample`.`inventory`.`landmark`(`city`)
CREATE INDEX `def_sourceairport` ON `travel-sample`(`sourceairport`)
...
```

Planung

In der Planungsphase legen Sie die Anforderungen an den Amazon DocumentDB-Cluster fest und ordnen die Couchbase-Buckets, -Bereiche und -Sammlungen den Amazon DocumentDB-Datenbanken und -Sammlungen zu.

Anforderungen an den Amazon DocumentDB-Cluster

Verwenden Sie die in der Discovery-Phase gesammelten Daten, um die Größe Ihres Amazon DocumentDB-Clusters zu bestimmen. Weitere Informationen zur Dimensionierung Ihres Amazon DocumentDB-Clusters finden Sie unter Instance-Größe.

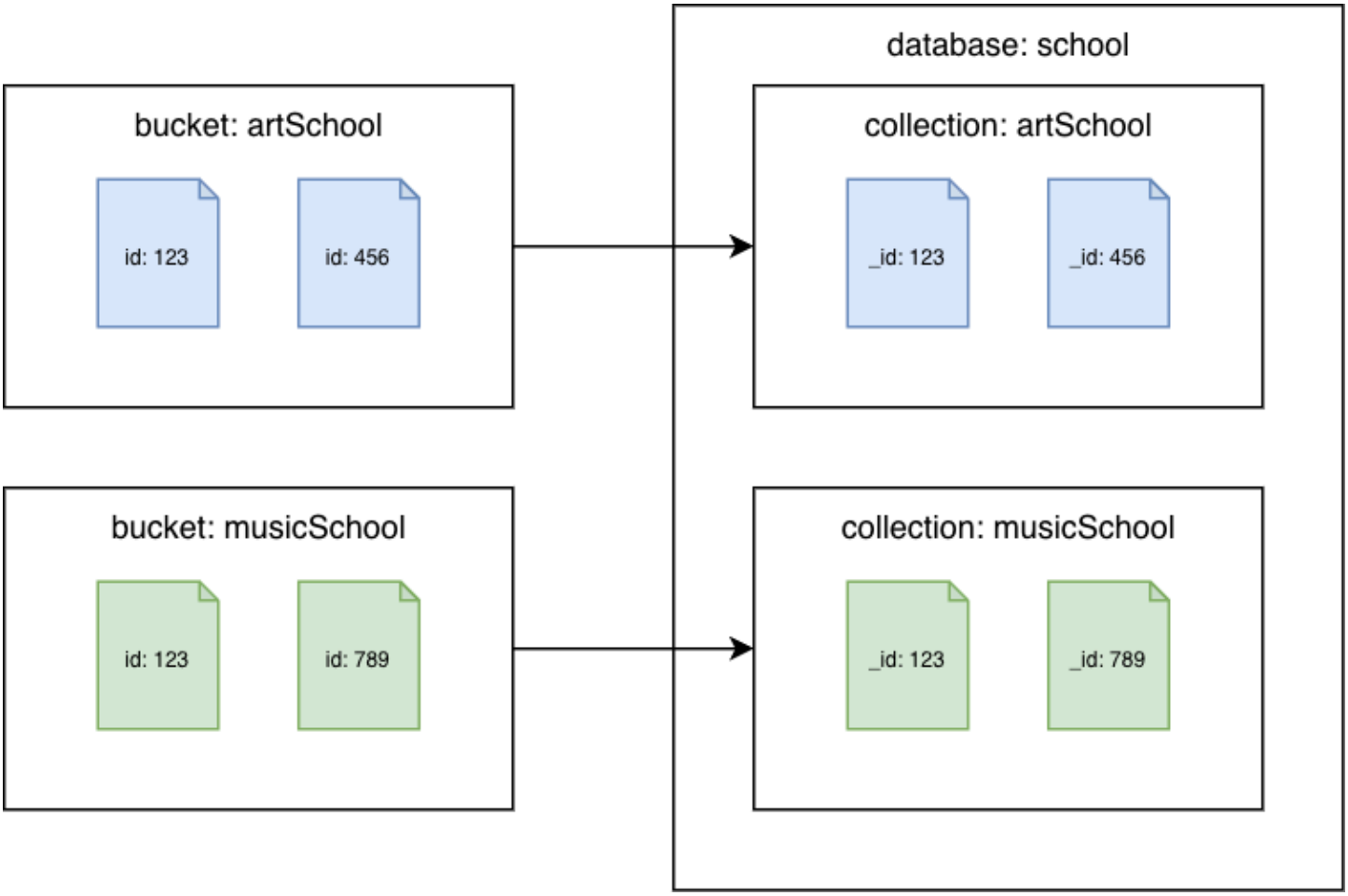
Zuordnung von Buckets, Bereichen und Sammlungen zu Datenbanken und Sammlungen

Ermitteln Sie die Datenbanken und Sammlungen, die in Ihren Amazon DocumentDB-Clustern existieren werden. Erwägen Sie die folgenden Optionen, je nachdem, wie die Daten in Ihrem Couchbase-Cluster organisiert sind. Dies sind nicht die einzigen Optionen, aber sie bieten Ansatzpunkte, die Sie in Betracht ziehen sollten.

Couchbase Server 6.x oder früher

Couchbase-Buckets für Amazon DocumentDB-Sammlungen

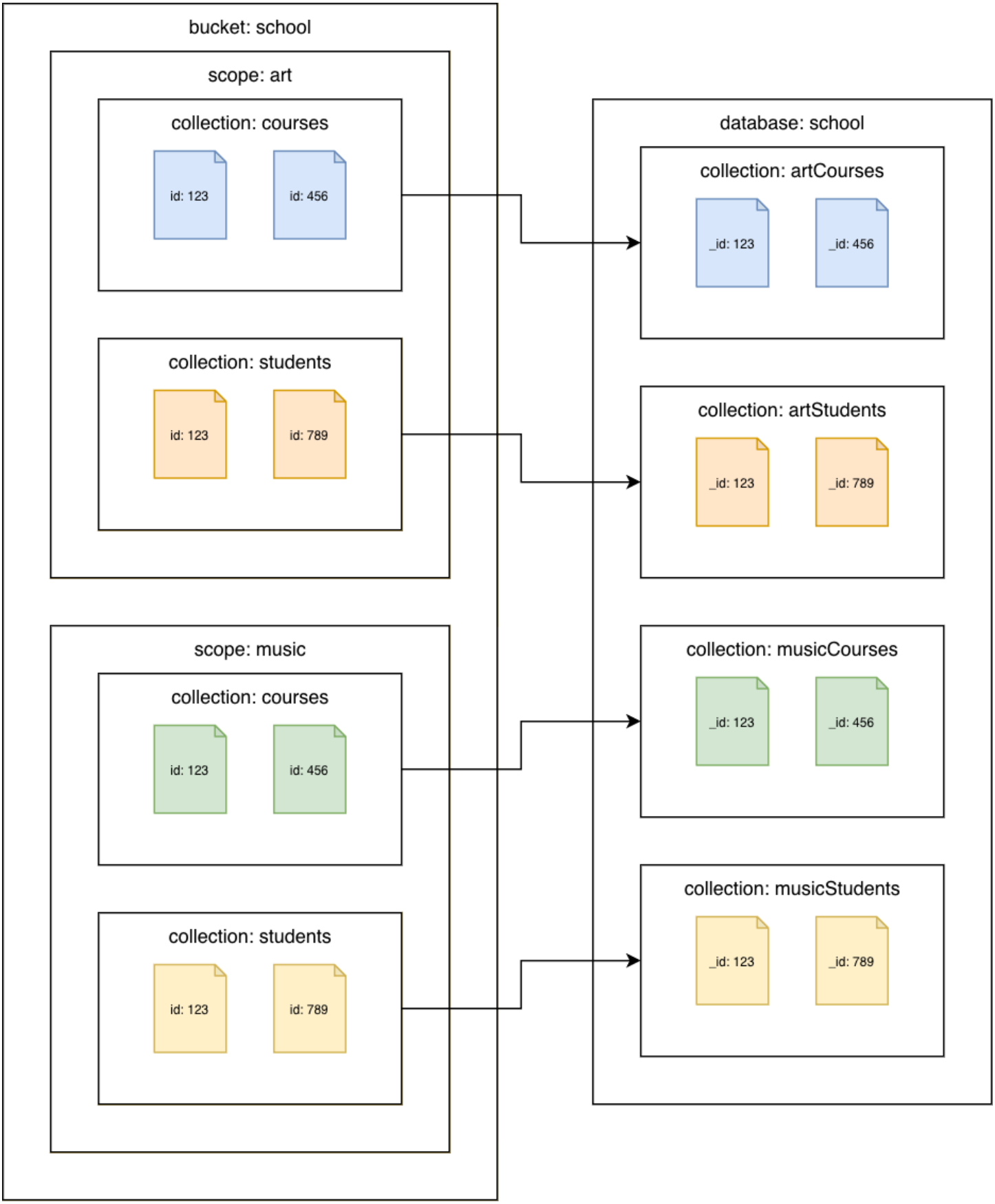
Migrieren Sie jeden Bucket in eine andere Amazon DocumentDB-Sammlung. In diesem Szenario wird der `id` Couchbase-Dokumentwert als Amazon DocumentDB-Wert verwendet. `_id`



Couchbase Server 7.0 oder höher

Couchbase-Sammlungen in Amazon DocumentDB-Sammlungen

Migrieren Sie jede Sammlung in eine andere Amazon DocumentDB-Sammlung. In diesem Szenario wird der `id` Couchbase-Dokumentwert als Amazon DocumentDB-Wert verwendet. `_id`


bucket: school
scope: art
collection: courses
id: 123
id: 456
collection: students
id: 123
id: 789
scope: music
collection: courses
id: 123
id: 456
collection: students
id: 123
id: 789
database: school
collection: artCourses
_id: 123
_id: 456
collection: artStudents
_id: 123
_id: 789
collection: musicCourses
_id: 123
_id: 456
collection: musicStudents
_id: 123
_id: 789

Migration

Migration des Indexes

Bei der Migration zu Amazon DocumentDB werden nicht nur Daten, sondern auch Indizes übertragen, um die Abfrageleistung aufrechtzuerhalten und den Datenbankbetrieb zu optimieren. In diesem Abschnitt wird der detaillierte, schrittweise Prozess zur Migration von Indizes zu Amazon DocumentDB beschrieben und gleichzeitig Kompatibilität und Effizienz gewährleistet.

Verwenden Sie Amazon Q (verfügbar in der AWS-Managementkonsole oder in Ihrer IDE), um `CREATE INDEX` SQL++-Anweisungen in Amazon DocumentDB-Befehle zu konvertieren. `createIndex()`

1. Laden Sie die <bucket name>Indexe.txt-Datei (en) hoch, die mit dem Discovery Tool for Couchbase erstellt wurden.
2. Geben Sie die folgende Eingabeaufforderung ein:

   ```
   Convert the Couchbase CREATE INDEX statements to Amazon DocumentDB
   createIndex commands
   ```

Amazon Q generiert entsprechende Amazon `createIndex()` DocumentDB-Befehle. Beachten Sie, dass Sie die Sammlungsnamen möglicherweise aktualisieren müssen, je nachdem, wie Sie die Couchbase-Buckets, -Bereiche und -Sammlungen den Amazon DocumentDB-Sammlungen zugeordnet haben.

Beispiel:

indexes-beer-sample.txt

```
CREATE INDEX `beerType` ON `beer-sample`(`type`)
CREATE INDEX `code` ON `beer-sample`(`code`) WHERE (`type` = "brewery")
```

Beispiel für eine Amazon Q-Ausgabe (Auszug):

```
db.beerSample.createIndex(
  { "type": 1 },
  {
    "name": "beerType",
    "background": true
  }
```

```
)

db.beerSample.createIndex(
  { "code": 1 },
  {
    "name": "code",
    "background": true,
    "partialFilterExpression": { "type": "brewery" }
  }
)
```

Weitere Informationen zu Indizes, die Amazon Q nicht konvertieren kann, finden Sie unter Amazon DocumentDB-Indizes und Indizes und Indexeigenschaften verwalten.

Refactor-Code zur Verwendung der MongoDB-APIs

Kunden verwenden die Couchbase SDKs, um eine Verbindung zum Couchbase Server herzustellen. Amazon DocumentDB-Clients verwenden MongoDB-Treiber, um eine Verbindung zu Amazon DocumentDB herzustellen. Alle von den Couchbase-SDKs unterstützten Sprachen werden auch von MongoDB-Treibern unterstützt. Weitere Informationen zum Treiber für Ihre Sprache finden Sie unter MongoDB-Treiber.

Da sich die APIs zwischen Couchbase Server und Amazon DocumentDB unterscheiden, müssen Sie Ihren Code umgestalten, um die entsprechenden MongoDB-APIs verwenden zu können. Sie können Amazon Q verwenden, um die K/V API-Aufrufe und SQL++-Abfragen in die entsprechenden MongoDB-APIs zu konvertieren:

1. Laden Sie die Quellcodedatei (en) hoch.
2. Geben Sie die folgende Eingabeaufforderung ein:

```
Convert the Couchbase API code to Amazon DocumentDB API code
```

Unter Verwendung des Python-Codebeispiels von Hello Couchbase generiert Amazon Q Folgendes (Auszug):

```
from pymongo import MongoClient

# Connection parameters
database_name = "travel-sample"

# Connect to Amazon DocumentDB cluster
```

```
client = MongoClient('<Amazon DocumentDB connection string>',
                     tls=True,
                     tlsCAFile='global-bundle.pem')

# Get reference to database and collection
db = client['travel-sample']
airline_collection = db['airline']

# upsert document function
def upsert_document(doc):
    print("\nUpsert Result: ")
    try:
        # key will equal: "airline_8091"
        key = doc["type"] + "_" + str(doc["id"])
        doc['_id'] = key # Amazon DocumentDB uses _id as primary key
        result = airline_collection.update_one(
            {'_id': key},
            {'$set': doc},
            upsert=True
        )
        print(f"Modified count: {result.modified_count}")
    except Exception as e:
        print(e)

# get document function
def get_airline_by_key(key):
    print("\nGet Result: ")
    try:
        result = airline_collection.find_one({'_id': key})
        print(result)
    except Exception as e:
        print(e)

# query for document by callsign
def lookup_by_callsign(cs):
    print("\nLookup Result: ")
    try:
        result = airline_collection.find(
            {'callsign': cs},
            {'name': 1, '_id': 0}
        )
        for doc in result:
            print(doc['name'])
    except Exception as e:
```

```
        print(e)

# Test document
airline = {
    "type": "airline",
    "id": 8091,
    "callsign": "CBS",
    "iata": None,
    "icao": None,
    "name": "Couchbase Airways",
}

upsert_document(airline)
get_airline_by_key("airline_8091")
lookup_by_callsign("CBS")
```

Beispiele für die Verbindung mit Amazon DocumentDB in Python, PHP, Go, Java, C#/.NET, Node.js R und Ruby finden Sie unter Programmgesteuertes Herstellen einer Verbindung zu Amazon DocumentDB.

Wählen Sie den Migrationsansatz

Bei der Migration von Daten zu Amazon DocumentDB gibt es zwei Optionen:

1. Offline-Migration
2. Online-Migration

> **Note**
>
> Amazon DocumentDB verwendet das `_id` Feld als Primärschlüssel für jedes Dokument. Die Migrationsansätze in diesem Handbuch speichern den Couchbase-Dokumentschlüssel im Feld. `_id` Bei der Offline-Migration erfolgt dies über die `--include-key _id` Option unter. `cbexport` Bei der Online-Migration erfolgt dies über den Kafka-Connector. `ProvidedInKeyStrategy`

Offline-Migration

Ziehen Sie eine Offline-Migration in Betracht, wenn:

- Ausfallzeiten sind akzeptabel: Bei der Offline-Migration werden Schreibvorgänge in die Quelldatenbank gestoppt, die Daten exportiert und anschließend in Amazon DocumentDB importiert. Dieser Prozess verursacht Ausfallzeiten für Ihre Anwendung. Wenn Ihre Anwendung oder Ihr Workload diesen Zeitraum der Nichtverfügbarkeit aushält, ist eine Offline-Migration eine praktikable Option.
- Migration kleinerer Datensätze oder Durchführung von Machbarkeitsstudien: Bei kleineren Datensätzen ist der Zeitaufwand für den Export- und Importvorgang relativ kurz, sodass die Offline-Migration eine schnelle und einfache Methode ist. Es eignet sich auch gut für Entwicklungs-, Test- und Machbarkeitsstudien, in denen Ausfallzeiten weniger kritisch sind.
- Einfachheit hat Priorität: Die Offline-Methode, die cbexport und mongoimport verwendet, ist im Allgemeinen der einfachste Ansatz zur Migration von Daten. Es vermeidet die Komplexität der Change Data Capture (CDC), die mit Online-Migrationsmethoden verbunden ist.
- Es müssen keine laufenden Änderungen repliziert werden: Wenn die Quelldatenbank während der Migration keine aktiven Änderungen empfängt oder wenn es nicht wichtig ist, dass diese Änderungen während des Migrationsprozesses erfasst und auf das Ziel angewendet werden, ist ein Offline-Ansatz angemessen.

Topics

- Couchbase Server 6.x oder früher
- Couchbase Server 7.0 oder höher

Couchbase Server 6.x oder früher

Sammlung vom Couchbase-Bucket zur Amazon DocumentDB-Sammlung

Exportieren Sie Daten mit cbexport json, um einen JSON-Dump aller Daten im Bucket zu erstellen. Für die `--format` Option können Sie oder verwenden. `lines` `list`

```
cbexport json \
  --cluster <source cluster endpoint> \
  --bucket <bucket name> \
  --format <lines | list> \
  --username <username> \
  --password <password> \
  --output export.json \
  --include-key _id
```

Importieren Sie die Daten mithilfe von Mongoimport in eine Amazon DocumentDB-Sammlung mit der entsprechenden Option zum Importieren der Zeilen oder Listen:

Zeilen:

```
mongoimport \
  --db <database> \
  --collection <collection> \
  --uri "<Amazon DocumentDB cluster connection string>" \
  --ssl \
  --sslCAFile global-bundle.pem \
  --file export.json
```

liste:

```
mongoimport \
  --db <database> \
  --collection <collection> \
  --uri "<Amazon DocumentDB cluster connection string>" \
  --ssl \
  --sslCAFile global-bundle.pem \
  --jsonArray \
  --file export.json
```

Couchbase Server 7.0 oder höher

Verwenden Sie die Tools cbexport und mongoimport, um eine Offline-Migration durchzuführen:

Couchbase-Bucket mit Standardumfang und Standardsammlung

Exportieren Sie Daten mit cbexport json, um einen JSON-Dump aller Sammlungen im Bucket zu erstellen. Für die `--format` Option können Sie oder verwenden. `lines` `list`

```
cbexport json \
  --cluster <source cluster endpoint> \
  --bucket <bucket name> \
  --format <lines | list> \
  --username <username> \
  --password <password> \
  --output export.json \
  --include-key _id
```

Importieren Sie die Daten mithilfe von Mongoimport in eine Amazon DocumentDB-Sammlung mit der entsprechenden Option zum Importieren der Zeilen oder Listen:

Zeilen:

```
mongoimport \
  --db <database> \
  --collection <collection> \
  --uri "<Amazon DocumentDB cluster connection string>" \
  --ssl \
  --sslCAFile global-bundle.pem \
  --file export.json
```

liste:

```
mongoimport \
  --db <database> \
  --collection <collection> \
  --uri "<Amazon DocumentDB cluster connection string>" \
  --ssl \
  --sslCAFile global-bundle.pem \
  --jsonArray \
  --file export.json
```

Couchbase-Sammlungen für Amazon DocumentDB-Sammlungen

Exportieren Sie Daten mit cbexport json, um einen JSON-Dump jeder Sammlung zu erstellen. Verwenden Sie die `--include-data` Option, um jede Sammlung zu exportieren. Für die `--format` Option können Sie `lines` oder verwenden`list`. Verwenden Sie die `--collection-field` Optionen `--scope-field` und, um den Namen des Bereichs und der Sammlung in den angegebenen Feldern in jedem JSON-Dokument zu speichern.

```
cbexport json \
  --cluster <source cluster endpoint> \
  --bucket <bucket name> \
  --include-data <scope name>.<collection name> \
  --format <lines | list> \
  --username <username> \
  --password <password> \
  --output export.json \
  --include-key _id \
  --scope-field "_scope" \
```

```
  --collection-field "_collection"
```

Da cbexport die `_collection` Felder `_scope` und zu jedem exportierten Dokument hinzugefügt hat, können Sie sie aus jedem Dokument in der Exportdatei mit Suchen und Ersetzen oder einer anderen von Ihnen `sed` bevorzugten Methode entfernen.

Importieren Sie die Daten für jede Sammlung mithilfe von mongoimport in eine Amazon DocumentDB-Sammlung mit der entsprechenden Option zum Importieren der Zeilen oder Listen:

Zeilen:

```
mongoimport \
  --db <database> \
  --collection <collection> \
  --uri "<Amazon DocumentDB cluster connection string>" \
  --ssl \
  --sslCAFile global-bundle.pem \
  --file export.json
```

liste:

```
mongoimport \
  --db <database> \
  --collection <collection> \
  --uri "<Amazon DocumentDB cluster connection string>" \
  --ssl \
  --sslCAFile global-bundle.pem \
  --jsonArray \
  --file export.json
```

Online-Migration

Ziehen Sie eine Online-Migration in Betracht, wenn Sie Ausfallzeiten minimieren müssen und laufende Änderungen nahezu in Echtzeit auf Amazon DocumentDB repliziert werden müssen.

Unter So führen Sie eine Live-Migration von Couchbase zu Amazon DocumentDB durch, um zu erfahren, wie Sie eine Live-Migration zu Amazon DocumentDB durchführen. Die Dokumentation führt Sie durch die Bereitstellung der Lösung und die Durchführung einer Live-Migration eines Buckets zu einem Amazon DocumentDB-Cluster.

Topics

- Couchbase Server 6.x oder früher
- Couchbase Server 7.0 oder höher

Couchbase Server 6.x oder früher

Sammlung vom Couchbase-Bucket zur Amazon DocumentDB-Sammlung

Das Migrationsprogramm für Couchbase ist vorkonfiguriert, um eine Online-Migration eines Couchbase-Buckets zu einer Amazon DocumentDB-Sammlung durchzuführen. Betrachtet man die Konfiguration des Senk-Connectors, so ist der `document.id.strategy` Parameter so konfiguriert, dass er den Nachrichtenschlüsselwert als `_id` Feldwert verwendet (siehe Eigenschaften der Sink-Connector-ID-Strategie):

```
ConnectorConfiguration:
  document.id.strategy:
 'com.mongodb.kafka.connect.sink.processor.id.strategy.ProvidedInKeyStrategy'
```

Couchbase Server 7.0 oder höher

Couchbase-Bucket mit Standardumfang und Standardsammlung

Das Migrationsprogramm für Couchbase ist vorkonfiguriert, um eine Online-Migration eines Couchbase-Buckets zu einer Amazon DocumentDB-Sammlung durchzuführen. Betrachtet man die Konfiguration des Senk-Connectors, so ist der `document.id.strategy` Parameter so konfiguriert, dass er den Nachrichtenschlüsselwert als `_id` Feldwert verwendet (siehe Eigenschaften der Sink-Connector-ID-Strategie):

```
ConnectorConfiguration:
  document.id.strategy:
 'com.mongodb.kafka.connect.sink.processor.id.strategy.ProvidedInKeyStrategy'
```

Couchbase-Sammlungen in Amazon DocumentDB-Sammlungen

Konfiguriere den Quell-Connector so, dass jede Couchbase-Sammlung in jedem Bereich zu einem separaten Thema gestreamt wird (siehe Quellkonfigurationsoptionen). Beispiel:

```
ConnectorConfiguration:
  # add couchbase.collections configuration
  couchbase.collections: '<scope 1>.<collection 1>, <scope 1>.<collection 2>, ...'
```

Konfigurieren Sie den Senk-Connector so, dass er von jedem Thema zu einer separaten Amazon DocumentDB-Sammlung streamt (siehe Konfigurationseigenschaften des Sink Connectors). Beispiel:

```
ConnectorConfiguration:
  # remove collection configuration
  #collection: 'test'

  # modify topics configuration
  topics: '<bucket>.<scope 1>.<collection 1>, <bucket>.<scope 1>.<collection 2>, ...'

  # add topic.override.%s.%s configurations for each topic
  topic.override.<bucket>.<scope 1>.<collection 1>.collection: '<collection>'
  topic.override.<bucket>.<scope 1>.<collection 2>.collection: '<collection>'
```

Validierung

Dieser Abschnitt enthält einen detaillierten Validierungsprozess zur Überprüfung der Datenkonsistenz und -integrität nach der Migration zu Amazon DocumentDB. Die Validierungsschritte gelten unabhängig von der Migrationsmethode.

Topics

- Stellen Sie sicher, dass alle Sammlungen im Ziel vorhanden sind
- Überprüfen Sie die Anzahl der Dokumente zwischen Quell- und Zielclustern
- Dokumente zwischen Quell- und Zielclustern vergleichen

Stellen Sie sicher, dass alle Sammlungen im Ziel vorhanden sind

Couchbase-Quelle

Option 1: Workbench abfragen

```
SELECT RAW `path`
  FROM system:keyspaces
  WHERE `bucket` = '<bucket>'
```

Option 2: CBQ-Tool

```
cbq \
  -e <source cluster endpoint> \
  -u <username> \
  -p <password> \
  -q "SELECT RAW `path`
       FROM system:keyspaces
       WHERE `bucket` = '<bucket>'"
```

Amazon DocumentDB-Ziel

mongosh (siehe Stellen Sie eine Verbindung zu Ihrem Amazon DocumentDB-Cluster her):

```
db.getSiblingDB('<database>')
db.getCollectionNames()
```

Überprüfen Sie die Anzahl der Dokumente zwischen Quell- und Zielclustern

Couchbase-Quelle

Couchbase Server 6.x oder früher

Option 1: Workbench abfragen

```
SELECT COUNT(*)
FROM `<bucket>`
```

Option 2: cbq https://docs.couchbase.com/server/current/cli/cbq-tool.html

```
cbq \
  -e <source cluster endpoint> \
  -u <username> \
  -p <password> \
  -q "SELECT COUNT(*)
       FROM `<bucket>`"
```

Couchbase Server 7.0 oder höher

Option 1: Workbench abfragen

```
SELECT COUNT(*)
FROM `<bucket>`.`<scope>`.`<collection>`
```

Option 2: cbq https://docs.couchbase.com/server/current/cli/cbq-tool.html

```
cbq \
  -e <source cluster endpoint> \
  -u <username> \
  -p <password> \
  -q "SELECT COUNT(*)
       FROM `<bucket>`.`<scope>`.`<collection>`"
```

Amazon DocumentDB-Ziel

mongosh (siehe Stellen Sie eine Verbindung zu Ihrem Amazon DocumentDB-Cluster her):

```
db = db.getSiblingDB('<database>')
db.getCollection('<collection>').countDocuments()
```

Dokumente zwischen Quell- und Zielclustern vergleichen

Couchbase-Quelle

Couchbase Server 6.x oder früher

Option 1: Workbench abfragen

```
SELECT META().id as _id, *
FROM `<bucket>`
LIMIT 5
```

Option 2: cbq https://docs.couchbase.com/server/current/cli/cbq-tool.html

```
cbq \
  -e <source cluster endpoint> \
  -u <username> \
```

```
  -p <password> \
  -q "SELECT META().id as _id, *
       FROM `<bucket>`
       LIMIT 5"
```

Couchbase Server 7.0 oder höher

Option 1: Workbench abfragen

```
SELECT META().id as _id, *
FROM `<bucket>`.`<scope>`.`<collection>`
LIMIT 5
```

Option 2: cbq https://docs.couchbase.com/server/current/cli/cbq-tool.html

```
cbq \
  -e <source cluster endpoint> \
  -u <username> \
  -p <password> \
  -q "SELECT META().id as _id, *
       FROM `<bucket>`.`<scope>`.`<collection>`
       LIMIT 5"
```

Amazon DocumentDB-Ziel

mongosh (siehe Stellen Sie eine Verbindung zu Ihrem Amazon DocumentDB-Cluster her):

```
db = db.getSiblingDB('<database>')
db.getCollection('<collection>').find({
  _id: {
    $in: [
      <_id 1>, <_id 2>, <_id 3>, <_id 4>, <_id 5>
    ]
  }
})
```

Unterstützungstermine der Amazon DocumentDB DocumentDB-Engine-Version

Sie können die folgenden Daten verwenden, um Ihre Test- und Upgrade-Zyklen zu planen.

Veröffentlichungskalender für Amazon DocumentDB DocumentDB-Hauptversionen

Amazon DocumentDB unterstützt derzeit die folgenden Hauptversionen. Informationen zum Upgrade Ihrer Engine-Version finden Sie unter Aktualisierung einer Hauptversion der Engine oderEin Upgrade durchführen mit AWS DMS.

Engine-Version	Datum der Veröffentlichung	Ende des Standard-Supports	Beginn des erweiterten Support (Preis für das erste Jahr)	Beginn des erweiterten Support (Preis für 3. Jahr)	Ende des erweiterten Support
Version 3.6	9. Januar 2019	30. März 2026	31. März 2026	31. März 2028	30. März 2029
Ausführung 4.0	9. November 2020	N/A	N/A	N/A	N/A
Ausführung 5.0	1. März 2023	N/A	N/A	N/A	N/A
Ausführung 8.0	14. November 2025	N/A	N/A	N/A	N/A

Amazon DocumentDB unterstützt mehrere Engine-Versionen im Rahmen der Standardunterstützung. Gegen eine Gebühr für erweiterten Support können Sie eine Version auch nach Ablauf des Standard-Supportdatums weiter ausführen. Weitere Informationen finden Sie unter Erweiterter Support für Amazon DocumentDB Amazon DocumentDB DocumentDB-Preise.

Veröffentlichungskalender für Amazon DocumentDB-Nebenversionen

Amazon DocumentDB unterstützt derzeit die folgenden Nebenversionen. Der Zeitplan für die Veröffentlichung hängt von zusätzlichen Funktionen oder Korrekturen ab. Bei Nebenversionen kann es sein, dass der Standardsupport vor den entsprechenden Hauptversionen ausläuft. Informationen zum Upgrade Ihrer Engine-Version finden Sie unterAktualisierung einer kleineren Engine-Version.

> **Note**
>
> Nebenversionen sind ab der Hauptversion 5.0 verfügbar.

Engine-Version	Datum der Veröffentlichung
Version 5.0.0 (LTS)	16. Februar 2026
Version 5.0.1	8. Juni 2026

Amazon DocumentDB bezeichnet bestimmte Versionen als Long-Term Support (LTS) -Versionen. Weitere Informationen finden Sie unter Long-Term Support- (LTS, Langzeit-Support) Versionen nutzen.

Direktes Upgrade der Hauptversion von Amazon DocumentDB

Sie können ein direktes Hauptversions-Upgrade (MVU) Ihres Amazon DocumentDB-Clusters durchführen und dabei dieselben Endpunkte, Speicher und Tags beibehalten. Ihre Anwendungen funktionieren weiterhin ohne Änderungen. Diese Funktion ist überall dort, AWS-Regionen wo Amazon DocumentDB verfügbar ist, ohne zusätzliche Kosten verfügbar.

> **Important**
>
> Ihr Cluster ist während des Upgrades nicht verfügbar und wird mehrmals neu gestartet. Stellen Sie nach dem Start des Upgrades keine Verbindung zum Cluster her, lesen Sie nicht aus dem Cluster und schreiben Sie nicht in den Cluster. Die Ausfallzeit variiert je nach Anzahl der Sammlungen, Indizes, Datenbanken und Instanzen. Führen Sie das Upgrade während des Wartungsfensters oder zu Zeiten mit geringer Auslastung durch.
> Nach dem Upgrade können Sie kein Downgrade auf eine frühere Version durchführen.
> Sie können Ihren Snapshot vor dem Upgrade bei Bedarf in einem neuen Cluster wiederherstellen.

Topics

- Unterstützte Upgrade-Pfade

- Voraussetzungen
- Bewährte Methoden
- Was ändert sich nach dem Upgrade auf Amazon DocumentDB 8.0
- Post-upgrade Überlegungen zu Clustern, die von 3.6 oder 4.0 aktualisiert wurden
- Durchführung des Upgrades
- Fehlerbehebung

Unterstützte Upgrade-Pfade

In der folgenden Tabelle sind die Ziel-Engine-Versionen aufgeführt, die Sie für jede Quellversion auswählen können. Es ist nicht erforderlich, zuerst auf die `.0` Nebenversion zu aktualisieren und dann ein Nebenversions-Upgrade durchzuführen.

Quellversion	Unterstützte Zielversionen
3.6	Jede veröffentlichte 5.0- oder 8.0-Nebenversion
4,0	Jede veröffentlichte 5.0- oder 8.0-Nebenversion
5.0.0	8.0.0, 8.0.1, 8.0.2
5.0.1	8.0.0, 8.0.1, 8.0.2
5.0.2	8.0.2

Überlegungen nach dem Upgrade finden Sie unter the section called “Post-upgrade Überlegungen zu Clustern, die von 3.6 oder 4.0 aktualisiert wurden” und. the section called “Was ändert sich nach dem Upgrade auf Amazon DocumentDB 8.0”

Verwenden Sie den AWS CLI Befehl`aws docdb describe-db-engine-versions`, um die in Ihrer AWS Region verfügbaren Nebenversionen anzuzeigen. Eine Liste der veröffentlichten Nebenversionen finden Sie unterVersionshinweise.

> **Note**
>
> In-place MVU wird für globale Cluster oder elastische Cluster nicht unterstützt. Um einen globalen Cluster zu aktualisieren, entfernen Sie die sekundären Cluster, konvertieren

Sie den primären Cluster in einen regionalen Cluster, führen Sie die MVU durch und erstellen Sie dann den globalen Cluster neu, indem Sie sekundäre Cluster mit denselben Namen hinzufügen, um Ihre Endpunkte beizubehalten. Während der aktualisierte primäre Server Daten auf die neuen sekundären Replikationen repliziert, fallen I/O Gebühren an. Die detaillierten Schritte finden Sie unter Einen Cluster aus einem globalen Amazon DocumentDB-Cluster entfernen.

Voraussetzungen

Important

Skalieren Sie Burstable-Instances vor dem Upgrade hoch. Wenn Ihr Cluster Burstable-Instance-Typen verwendet (z. B. `db.t3.medium` oder`db.t4g.medium`), skalieren Sie die primäre Instance auf mindestens `db.r5.large` oder `db.r6g.large` bevor Sie das Upgrade initiieren. Burstable-Instances verfügen möglicherweise nicht über genügend CPU und Arbeitsspeicher, um den Upgrade-Vorgang abzuschließen. Dies kann zu Upgrade-Fehlern und einer längeren Nichtverfügbarkeit des Clusters führen. Sie können nach Abschluss des Upgrades wieder herunterskalieren.

Skalieren Sie die serverlose Writer-Instanz vor dem Upgrade hoch. Wenn Ihr Cluster über eine serverlose Writer-Instance verfügt, `ServerlessV2ScalingConfiguration` muss die maximale DCU in der auf 3 oder höher gesetzt werden, bevor das Upgrade initiiert wird. Cluster mit einer maximalen DCU unter 3 können nicht aktualisiert werden, da sie nicht über genügend CPU oder Arbeitsspeicher verfügen, um das Upgrade abzuschließen.

Überprüfen Sie Ihre Teilindizes vor dem Upgrade (Upgrade von Amazon DocumentDB 5.0 auf 8.0). Bevor Sie mit dem Upgrade beginnen, überprüfen Sie Ihre Teilindizes auf einen`partialFilterExpression`, der den `$type` Operator verwendet, um bestimmte BSON-Typen abzugleichen: insbesondere die Aliase `binData` `bool` `datedouble`,`decimal`,, und den `array` Typ oder einen numerischen Typcode wie. `{ "$type" : 1 }` Solche Teilindizes führen dazu, dass das Upgrade fehlschlägt, obwohl der Index für Lese- und Schreibvorgänge normal funktioniert. Verwenden Sie diese `db.collection.getIndexes()` Option, um Ihre `partialFilterExpression` Teilindizes zu überprüfen. Löschen Sie den betroffenen Teilindex, bevor Sie das Upgrade starten, und erstellen Sie ihn nach Abschluss des Upgrades erneut. Wenn Sie sich nicht sicher sind, ob ein Index betroffen ist, wenden Sie sich vor dem Upgrade an den AWS Support.

- Instanztyp — Amazon DocumentDB 4.0+ unterstützt keine db.r4-Instances. Ändern Sie vor dem Upgrade alle `db.r4.*` Instances in Instances oder neuere `db.r5.*` Versionen. Siehe Ändern einer Amazon DocumentDB DocumentDB-Instance und Unterstützte Instance-Klassen nach Region.
- Betriebssystem-Patches — Wenden Sie vor dem Upgrade alle ausstehenden Wartungsmaßnahmen für das Betriebssystem auf alle Instanzen an. Siehe Aktualisierungen des Amazon DocumentDB-Betriebssystems.

> **Note**
>
> Ausstehende Engine-Patches auf Clusterebene können Instanz-Betriebssystem-Patches verbergen. Wenden Sie bei Bedarf zuerst Engine-Patches an. Siehe Durchführung eines Patch-Updates für die Engine-Version eines Clusters.

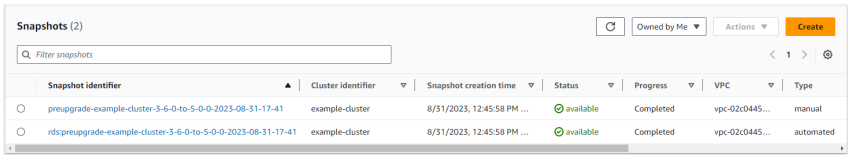
- Indexgrenzwerte für Burstable-Instances (Instances der T-Familie) — Wenn Sie mehr als 3.000 Indizes auf Burstable-Instances haben, skalieren Sie vor dem Upgrade die primäre Instanz auf mindestens db.r5.large. Sie können nach Abschluss des Upgrades wieder herunterskalieren.

Instance	Maximale Indizes für MVU
db.t4g.medium	3K
db.t3.medium	10 K

- Parametergruppe — Halten Sie vor dem Upgrade eine benutzerdefinierte Cluster-Parametergruppe für die Zielversion bereit. Wenn keine Parametergruppe angegeben ist, wird die Standardparametergruppe für die Zielversion verwendet (z. B. `default.docdb5.0` oder`default.docdb8.0`).
- Manueller Snapshot — Erstellen Sie vor dem Upgrade einen manuellen Snapshot. Beim Upgrade-Vorgang wird ein automatischer Snapshot mit dem Namen `preupgrade-<name>-<version>-<timestamp>` erstellt. Erstellen Sie jedoch immer Ihr eigenes Backup. Siehe Einen manuellen Cluster-Snapshot erstellen.

> **Note**
>
> Der durch den Upgrade-Vorgang erstellte automatische Snapshot wird nicht automatisch gelöscht, nachdem das direkte Upgrade der Hauptversion abgeschlossen ist. Für diesen Snapshot fallen keine Gebühren an, solange er innerhalb der Aufbewahrungsfrist liegt. Sie

können sich dafür entscheiden, diesen Snapshot zu löschen, sobald Sie ein erfolgreiches Upgrade Ihres Clusters überprüft haben.



Bewährte Methoden

Vor dem Upgrade — Testen mit einem Klon

1. Wird verwendetKlonen eines Volumes für einen Amazon DocumentDB-Cluster, um einen Klon Ihres Clusters zu erstellen. Es fallen keine Speicherkosten an, es sei denn, Sie ändern die Daten auf dem Clone.
2. Ordnen Sie die Anzahl der Instanzen des Clones dem Ziel-Cluster zu, um eine realistische Zeitschätzung zu erhalten.
3. Führen Sie die MVU auf dem Clone durch und testen Sie sie vollständig auf funktionale Unterschiede.
4. Prüfen Sie, ob ein Upgrade bereits geplant ist, indem Sie es ausführen `describe-db-clusters` und danach suchen`PendingModifiedValues.EngineVersion`. Wenn Sie den Cluster geändert und ausgewählt haben, dass er im nächsten Wartungsfenster angewendet werden soll, ist der Zeitplan in der Konsole nicht sichtbar. Sie können ihn jedoch im folgenden Ordner einsehen AWS CLI:

```
aws docdb describe-db-clusters \
  --region us-east-1 \
  --db-cluster-identifier mydocdbcluster
```

```
"PendingModifiedValues": {
    "EngineVersion": "5.0.0"
},
```

5. Wenn der Test erfolgreich ist, fahren Sie mit dem Upgrade auf Ihrem Produktionscluster fort.

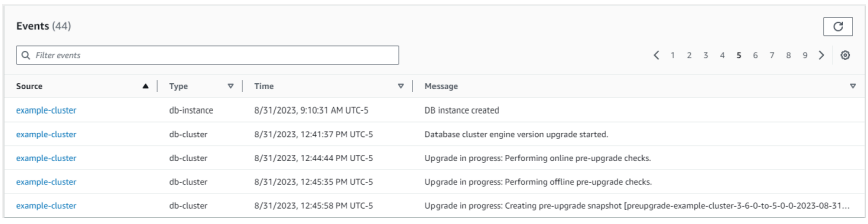
Während des Upgrades

Sie können den Fortschritt Ihres direkten Hauptversions-Upgrades überwachen, indem Sie Cluster-Wartungsveranstaltungen abonnieren. Wenn das Upgrade abgeschlossen ist, erhalten Sie das Ereignis „Die Hauptversion des Datenbank-Clusters wurde aktualisiert“. Dieses und andere Ereignisse, die während des Upgrades auftreten, werden im Abschnitt Ereignisse und Tags der Cluster-Detailseite in der Amazon DocumentDB-Konsole angezeigt. Der Cluster-Status ändert sich dann von `upgrading` zu`available`.

Die folgenden Ereignisse werden während des Upgrades generiert:

1. Das Upgrade der Hauptversion der Datenbankcluster-Engine wurde gestartet. Der Cluster bleibt online.
2. Upgrade-Vorbereitung im Gange: Die Online-Upgrade-Vorabprüfungen werden gestartet.
3. Upgrade-Vorbereitung im Gange: Die Online-Upgrade-Vorabprüfungen wurden abgeschlossen.
4. Der Datenbank-Cluster wird offline geschaltet, während die primäre Instanz den patch/upgrade Vorgang abschließt.
5. Upgrade-Vorbereitung läuft: Offline-Upgrade-Vorabprüfungen werden gestartet.
6. Upgrade-Vorbereitung läuft: Offline-Upgrade-Vorabprüfungen wurden abgeschlossen.
7. <version-to><timestamp>Upgrade wird ausgeführt: Vor dem Upgrade wird ein Snapshot erstellt [preupgrade- <cluster-name>- <version-from>-to- -].
8. Upgrade wird ausgeführt: Volume wird geklont.
9. Aktualisierung im Gange: Writer wird aktualisiert.
10. Aktualisierung im Gange: Reader werden aktualisiert.
11. Die Hauptversion der Datenbank-Cluster-Engine wurde aktualisiert.

Ereignisse sind in der Konsole auf der Seite „Ereignisse“ sichtbar:



Von dort aus können Sie das Programm ausführen AWS CLI, [`aws docdb describe-events`] um den Upgrade-Fortschritt zu überwachen. Um Benachrichtigungen automatisch zu

erhalten, abonnieren Sie Ereignisse und leiten Sie sie für E-Mail, Push-Nachrichten oder andere Übermittlungsmethoden an Amazon SNS weiter. `aws docdb create-event-subscription` Weitere Informationen finden Sie unter Amazon DocumentDB DocumentDB-Veranstaltungen abonnieren.

```
aws docdb describe-events
  --source-identifier mydocdbcluster
  --source-type db-cluster
```

Daraufhin erhalten Sie ein Ergebnis, das dem hier dargestellten entspricht:

```
{
    "Events": [
        {
            "SourceIdentifier": "mydocdbcluster",
            "SourceType": "db-cluster",
            "Message": "Database cluster engine version upgrade started.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2023-07-11T23:20:32.444000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:xxxx:cluster:mycluster"
        }
    ]
}
```

Nach dem Upgrade

Warning

Unmittelbar nach dem direkten Upgrade der Hauptversion füllt Ihr Amazon DocumentDB-Cluster die Index-Metadaten erneut auf, die die Datenbank-Engine zur Optimierung der Abfrageausführungspläne verwendet. Sobald dieser Vorgang abgeschlossen ist, kehrt die Abfrageleistung auf das erwartete Niveau zurück. In der Regel ist er in wenigen Minuten abgeschlossen, kann aber je nach Anzahl der Indizes in Ihrem Cluster bis zu zwei Stunden dauern.

Starten Sie up/down Ihre Writer-Instance während dieser Zeit nicht neu, führen Sie kein Failover durch oder skalieren Sie sie nicht, da dies die Neuberechnung der Index-Metadaten stören kann. Warten Sie, bis Sie die erwartete Abfrageleistung beobachten, bevor Sie solche Änderungen vornehmen.

Verfolgen Sie den Fortschritt anhand der folgenden Cluster-Ereignisse:

- Post-upgrade Cluster-Status: Die Aktualisierung der Index-Metadaten wurde gestartet
- Post-upgrade Clusterstatus: Die Aktualisierung der Indexmetadaten wurde in X Sekunden abgeschlossen

Wenden Sie sich an den AWS Support, wenn die Aktualisierung der Index-Metadaten nicht innerhalb von drei Stunden abgeschlossen ist oder wenn nach Abschluss des Vorgangs weiterhin Leistungsprobleme auftreten.

1. Erstellen Sie einen manuellen Snapshot des aktualisierten Clusters für den Fall, dass Sie den Zustand nach dem Upgrade wiederherstellen müssen. Der automatische Snapshot-Vorgang wird fortgesetzt, sobald das direkte Upgrade der Hauptversion abgeschlossen ist.
2. Tag-Cluster wurden von Version 3.6 aktualisiert. Fügen Sie ein Tag hinzu, um Cluster zu unterscheiden, für die ein Upgrade von 3.6 durchgeführt wurde. (siehethe section called “Post-upgrade Überlegungen zu Clustern, die von 3.6 oder 4.0 aktualisiert wurden”).
3. Aktualisieren Sie Ihren Treiber. Um neue Funktionen zu verwenden (z. B. Sortierung, Ansichten oder Zstd-Komprimierung in 8.0), führen Sie ein Upgrade auf die entsprechende MongoDB-API-Version durch. Weitere Informationen finden Sie unter Was ist neu in Amazon DocumentDB 8.0.
4. Gründlich testen. Validieren Sie Ihre Anwendung anhand des aktualisierten Clusters.

Was ändert sich nach dem Upgrade auf Amazon DocumentDB 8.0

Nach einem Upgrade der Hauptversion auf Amazon DocumentDB 8.0 (von Amazon DocumentDB 5.0 oder direkt von 3.6 oder 4.0) werden die folgenden Funktionen aktiviert oder geändert:

- Sortierung. Amazon DocumentDB 8.0 unterstützt die Sortierung. Nach dem Upgrade ist die Sortierung für neue Sammlungen und ihre Indizes sowie für neue Indizes vorhandener Sammlungen standardmäßig aktiviert.
- Textindex. Neue Textindizes werden mit Text Index V2 erstellt, der einen aktualisierten Textsuchparser für eine verbesserte MongoDB-Kompatibilität verwendet. Bestehende Textindizes sind nicht betroffen.
- Version des Abfrageplaners. Wenn Sie keine benutzerdefinierte Parametergruppe hatten, wird eine neue Standardparametergruppe für Amazon DocumentDB 8.0 erstellt, wobei Planner Version

3 automatisch ausgewählt wird. In der Version 3 des Abfrageplaners sind auch Ansichten verfügbar.

- Komprimierung. Amazon DocumentDB 8.0 unterstützt die wörterbuchbasierte Dokumentenkomprimierung mithilfe des Zstd-Algorithmus. Nach dem Upgrade werden neue Sammlungen mit standardmäßig aktivierter Zstd-Komprimierung erstellt. Bestehende Sammlungen aus 5.0 behalten ihre Komprimierungseinstellungen bei. Um die Vorteile der Zstd-Komprimierung für bestehende Sammlungen zu nutzen, können Sie deren Komprimierungseinstellungen ändern. Weitere Informationen finden Sie unter Verwaltung der wörterbuchbasierten Komprimierung in Amazon DocumentDB 8.0.
- Neuerstellung des Indexes Wenn Sie ein Upgrade von Amazon DocumentDB 5.0 auf Amazon DocumentDB 8.0 durchführen, ist keine Indexneuerstellung erforderlich. Wenn Sie direkt von Amazon DocumentDB 3.6 oder 4.0 auf 8.0 aktualisieren, erstellen Sie Ihre Indizes neu, wie unter beschrieben. the section called “Post-upgrade Überlegungen zu Clustern, die von 3.6 oder 4.0 aktualisiert wurden”

> **Important**
>
> Amazon DocumentDB 8.0 erfordert TLS 1.2 oder höher. TLS 1.0 und TLS 1.1 werden nicht mehr unterstützt.

> **Note**
>
> Eine vollständige Liste der funktionalen Unterschiede finden Sie unterAmazon DocumentDB-Kompatibilität mit MongoDB.

Post-upgrade Überlegungen zu Clustern, die von 3.6 oder 4.0 aktualisiert wurden

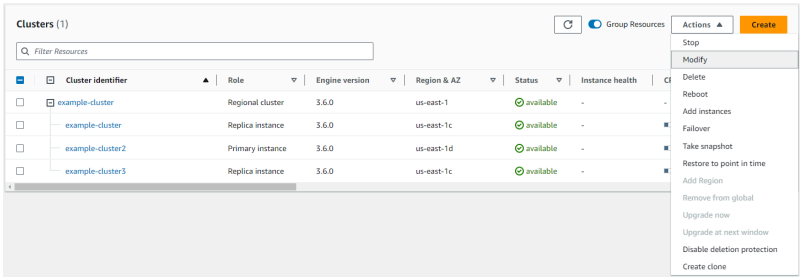
- Neuerstellung des Indexes Eine MVU behält die ursprünglichen Indizes bei. Amazon DocumentDB 5.0 und höher haben die Indexverwaltung und die Speicherbereinigung verbessert, insbesondere für Indizes mit niedriger Kardinalität. Erstellen Sie nach dem Upgrade von 3.6 oder 4.0 (entweder 5.0 oder 8.0) Ihre Indizes neu, um eine optimale Abfrageleistung sicherzustellen (optional, erfordert zusätzliche). I/O Siehe Indexverwaltung mithilfe von `Reindex`.

- Numerischer Vergleich von Unterdokumenten (nur 3.6). Cluster, die von 3.6 aktualisiert wurden, erben das 3.6-Verhalten, bei dem numerische Typen in Unterdokumenten nicht typübergreifend verglichen werden. `{a: {b: NumberLong(1)}}`Entspricht beispielsweise `{a: {b: 1}}` in 3.6 nicht, aber sie sind in 4.0 und höher gleich. Dieses Verhalten wirkt sich auf alle Cluster aus, die von 3.6 aktualisiert wurden.

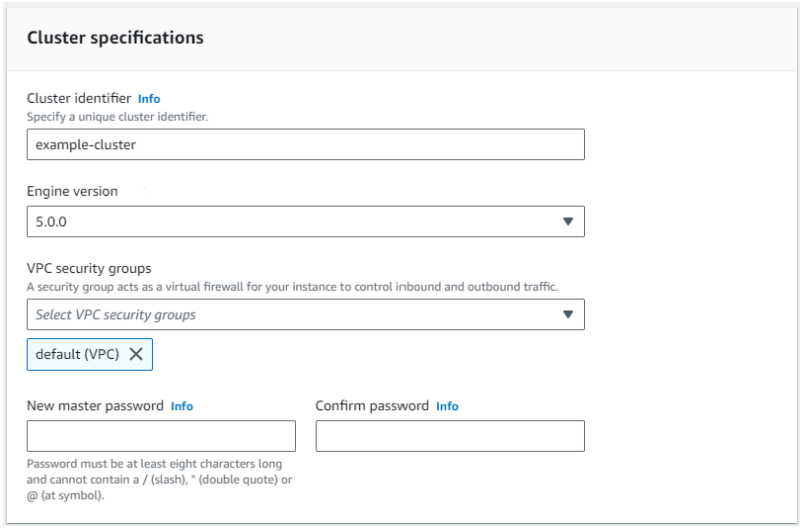
Durchführung des Upgrades

Using the AWS-Managementkonsole

1. Melden Sie sich bei der Amazon DocumentDB-Konsole an AWS-Managementkonsole und öffnen Sie sie.
2. Wählen Sie in der Cluster-Tabelle den Quell-Cluster aus, wählen Sie Actions und anschließend Modify aus.

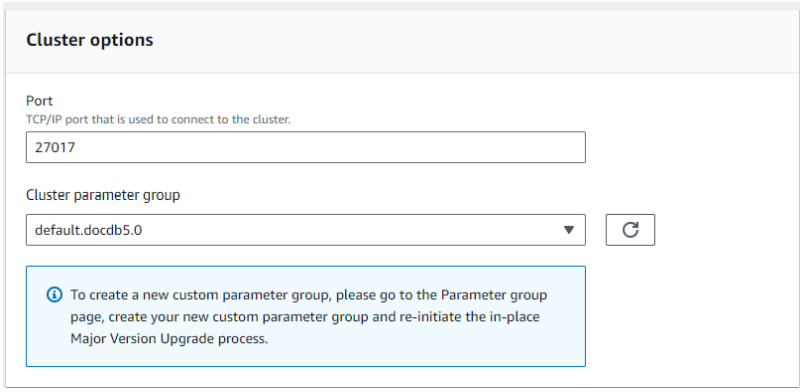


3. Wählen Sie unter Cluster-Spezifikationen die Zielversion (z. B. 5.0.0 oder 8.0.0) aus dem Drop-down-Menü Engine-Version aus.

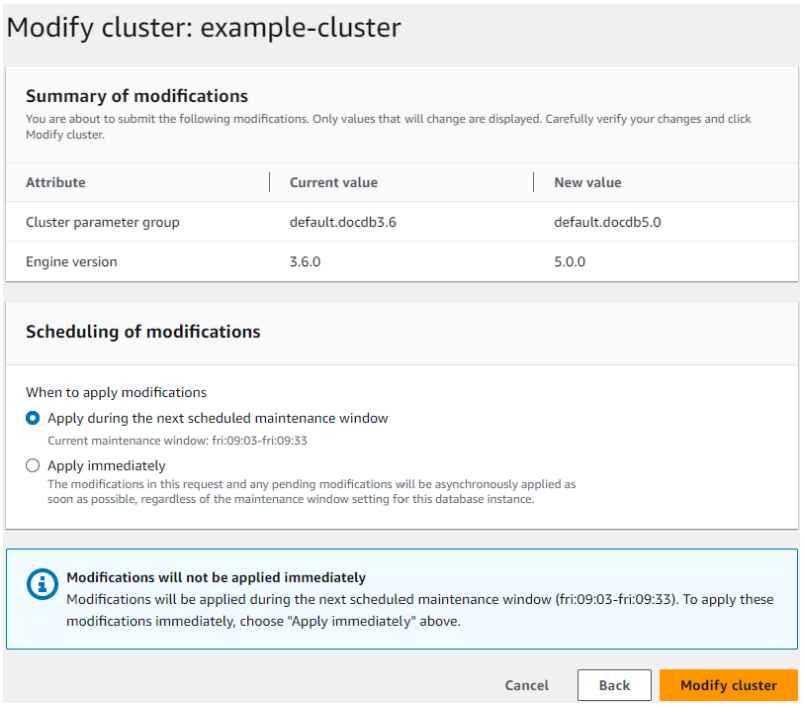


4. Wählen Sie in den Cluster-Optionen Ihre Cluster-Parametergruppe für die Ziel-Engine-Version aus. Sie können die Standardparametergruppe (z. B. default.docdb5.0 oder

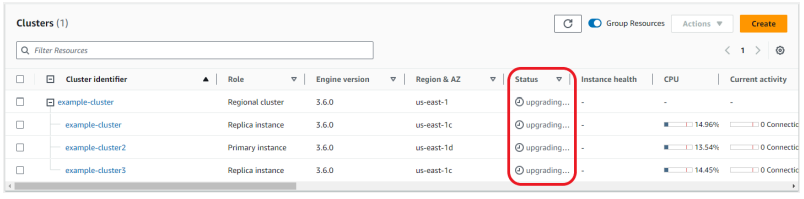
default.docdb8.0) oder eine benutzerdefinierte Parametergruppe verwenden, die Sie erstellt haben.



5. Wählen Sie „Fortfahren“, wählen Sie Ihre bevorzugte Terminplanung (sofort anwenden oder nächstes Wartungsfenster) und wählen Sie dann „Cluster ändern“.



6. Überwachen Sie den Clusterstatus in der Cluster-Tabelle, wenn er sich beim Upgrade ändert:



Using the AWS CLI

`modify-db-cluster`Mit der `--allow-major-version-upgrade` Flagge verwenden:

```
aws docdb modify-db-cluster \
  --db-cluster-identifier mydocdbcluster \
  --allow-major-version-upgrade \
  --engine-version 8.0.0 \
  --apply-immediately \
  --cluster-parameter-group mydocdbparametergroup \
  --region us-east-1
```

Ersetzen Sie jedes *placeholder* durch die Informationen Ihres Clusters.

Fehlerbehebung

- Pre-upgrade Überprüfen Sie den Fehler. Bevor das Upgrade beginnt, führt Amazon DocumentDB Validierungsprüfungen vor dem Upgrade durch. Die folgenden Ursachen sind häufig für Fehlschläge bei der Vorabprüfung aufgeführt:
  - Das Upgrade konnte nicht fortgesetzt werden, da die Sammlung (en) Namen mit 58 oder mehr Zeichen haben. Benennen Sie die betroffenen Sammlungen in kürzere Namen um, bevor Sie das Upgrade erneut versuchen.
  - Das Upgrade konnte nicht fortgesetzt werden, da die Indexanzahl den Grenzwert für den Instanztyp überschreitet. Führen Sie ein Upgrade auf einen größeren Instanztyp durch, bevor Sie das Upgrade erneut versuchen. Informationen zu Indexlimits nach Instanztyp finden Sie unterVoraussetzungen.
- Upgrade-Fehler und Rollback. Schlägt das Upgrade fehl, wird automatisch ein Rollback versucht. Ein erfolgreiches Rollback generiert das folgende Ereignis: „Der Datenbankcluster befindet sich in einem Zustand, der nicht aktualisiert werden kann.“ Ihr Cluster kehrt in den Zustand vor dem Upgrade zurück und Sie können ihn weiter verwenden. Wenden Sie sich zur Problembehebung an den AWS Support, bevor Sie es erneut versuchen.
- Post-upgrade Leistung. Während der Aktualisierung der Indexmetadaten können vorübergehende Leistungseinbußen und eine hohe CPU-Auslastung auftreten. Wenn die Leistungseinbußen länger als 3 Stunden andauern, wenden Sie sich an den AWS Support.

Für weitere Unterstützung wenden Sie sich bitte an AWS Support.

Upgrade der Nebenversion von Amazon DocumentDB

> **Note**
>
> In diesem Thema werden kleinere Versionsupgrades innerhalb derselben Hauptversion behandelt (z. B. das Upgrade von 5.0.0 auf 5.0.1). Informationen zum Upgrade auf eine andere Hauptversion finden Sie unter. Direktes Upgrade der Hauptversion von Amazon DocumentDB

Sie können die Minor-Engine-Version Ihres Amazon DocumentDB-Clusters aktualisieren (z. B. von 5.0.0 auf 5.0.1), indem Sie den Cluster ändern. Upgrades für kleinere Versionen können sofort oder während des nächsten Wartungsfensters angewendet werden.

Vor dem Durchführen eines Nebenversions-Upgrades

Bevor Sie ein Upgrade auf eine kleinere Version durchführen, können die folgenden Aktionen hilfreich sein:

- Führen Sie das Upgrade in Zeiten geringer Auslastung durch, um die Auswirkungen kurzzeitiger Ausfallzeiten zu verringern.
- Stellen Sie mithilfe des `describe-db-engine-versions` Befehls sicher, dass es sich bei der Ziel-Nebenversion um ein gültiges Upgrade-Ziel für Ihre aktuelle Version handelt.
- Bei globalen Clustern müssen die sekundären Cluster vor dem primären Cluster aktualisiert werden. Weitere Informationen finden Sie unter Patchen globaler Cluster.

> **Important**
>
> - Upgrades sind unidirektionale Operationen — Sie können nach dem Upgrade kein Downgrade mehr durchführen.
> - Bei Ihrem Cluster kommt es während des Upgrade-Vorgangs zu kurzen Ausfallzeiten.

Durchführen eines Nebenversions-Upgrades

Using the AWS-Managementkonsole

Um ein Upgrade einer Nebenversion durchzuführen, verwenden Sie AWS-Managementkonsole:

1. Melden Sie sich bei der Amazon DocumentDB DocumentDB-Konsole an AWS-Managementkonsoleund öffnen Sie sie.
2. Wählen Sie in der Cluster-Tabelle den Cluster aus, für den Sie ein Upgrade durchführen möchten, klicken Sie auf Aktionen und dann auf Ändern.
3. Wählen Sie im Dialogfeld Cluster ändern im Abschnitt Clusterspezifikationen die Ziel-Nebenversion aus dem Dropdownmenü Engine-Version aus.
4. Scrollen Sie nach unten und wählen Sie Weiter.
5. Wählen Sie im Abschnitt Planung von Änderungen aus, ob die Änderung sofort oder während des nächsten Wartungsfensters übernommen werden soll.

   Wählen Sie Cluster ändern aus.

Using the AWS CLI

Verwenden Sie den `modify-db-cluster`Befehl mit dem `--engine-version` Parameter:

```
aws docdb modify-db-cluster \
    --db-cluster-identifier mydocdbcluster \
    --engine-version 5.0.1 \
    --apply-immediately \
    --region us-east-1
```

Ersetzen Sie im obigen Beispiel jeden *`user input placeholder`* durch die Informationen Ihres Clusters.

> **Note**
>
> Wenn Sie das Upgrade nicht sofort, sondern im nächsten Wartungsfenster anwenden möchten, verwenden Sie `--no-apply-immediately` anstelle von`--apply-immediately`.

Das Upgrade der Nebenversion wird überprüft

Überprüfen Sie nach Abschluss des Upgrades die Engine-Version mit dem AWS CLI:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier mydocdbcluster \
    --query 'DBClusters[0].EngineVersion'
```

Die Ausgabe dieser Operation sieht in etwa wie folgt aus:

```
"5.0.1"
```

Aktualisieren Ihres Amazon DocumentDB-Clusters mit AWS Database Migration Service

> **Important**
>
> Amazon DocumentDB folgt nicht denselben Support-Lebenszyklen wie MongoDB, und der Zeitplan für das Ende der Nutzungsdauer von MongoDB gilt nicht für Amazon DocumentDB.

Sie können Ihren Amazon DocumentDB-Cluster mit minimaler Ausfallzeit auf eine höhere Version aktualisieren, indem Sie AWS DMS. AWS DMS ist ein vollständig verwalteter Service, der die Migration von älteren Amazon DocumentDB DocumentDB-Versionen, relationalen Datenbanken und nicht-relationalen Datenbanken zu Ihrem Amazon DocumentDB DocumentDB-Zielcluster erleichtert.

Topics

- Schritt 1: Change Streams aktivieren
- Schritt 2: Ändern Sie die Aufbewahrungsdauer für Change-Streams
- Schritt 3: Migrieren Sie Ihre Indizes
- Schritt 4: Erstellen Sie ein AWS DMS replication instance
- Schritt 5: Erstellen Sie eine AWS DMS Quell-Endpunkt
- Schritt 6: Erstellen Sie eine AWS DMS Zielendpunkt
- Schritt 7: Erstellen Sie eine Migrationsaufgabe und führen Sie sie aus

- Schritt 8: Ändern des Anwendungsendpunkts auf den Amazon DocumentDB DocumentDB-Zielcluster

Schritt 1: Change Streams aktivieren

Um eine Migration mit minimalen Ausfallzeiten durchzuführen, ist Zugriff auf die Change-Streams des Clusters AWS DMS erforderlich. Amazon DocumentDB DocumentDB-Change-Streams bieten eine zeitlich geordnete Abfolge von Aktualisierungsereignissen, die in den Sammlungen und Datenbanken Ihres Clusters auftreten. Das Lesen aus dem Change-Stream ermöglicht es AWS DMS , Change Data Capture (CDC) durchzuführen und inkrementelle Aktualisierungen auf den Amazon DocumentDB DocumentDB-Zielcluster anzuwenden.

Um Change-Streams für alle Sammlungen in einer bestimmten Datenbank zu aktivieren, authentifizieren Sie sich mit der Mongo-Shell bei Ihrem Amazon DocumentDB-Cluster und führen Sie die folgenden Befehle aus:

```
db.adminCommand({modifyChangeStreams: 1,
    database: "db_name",
    collection: "",
    enable: true});
```

Schritt 2: Ändern Sie die Aufbewahrungsdauer für Change-Streams

Ändern Sie als Nächstes den Aufbewahrungszeitraum für den Change-Stream je nachdem, wie lange Sie Änderungsereignisse im Change-Stream behalten möchten. Wenn Sie beispielsweise davon ausgehen, dass Ihre Amazon AWS DMS DocumentDB-Cluster-Migration 12 Stunden dauern wird, sollten Sie die Aufbewahrung des Change-Streams auf einen Wert von mehr als 12 Stunden festlegen. Die Standardaufbewahrungsdauer für Ihren Amazon DocumentDB-Cluster beträgt drei Stunden. Sie können die Aufbewahrungsdauer des Change-Stream-Protokolls für Ihren Amazon DocumentDB-Cluster zwischen einer Stunde und sieben Tagen ändern, indem Sie die AWS-Managementkonsole oder die AWS CLI verwenden. Weitere Informationen finden Sie unter Änderung der Aufbewahrungsdauer des Change-Stream-Protokolls.

Schritt 3: Migrieren Sie Ihre Indizes

Erstellen Sie dieselben Indizes auf Ihrem Amazon DocumentDB DocumentDB-Zielcluster wie auf Ihrem Amazon DocumentDB DocumentDB-Quellcluster. Es AWS DMS wickelt zwar die Migration

von Daten ab, migriert aber keine Indizes. Um die Indizes zu migrieren, verwenden Sie das Amazon DocumentDB Index Tool, um Indizes aus dem Amazon DocumentDB DocumentDB-Quellcluster zu exportieren. Sie können das Tool herunterladen, indem Sie einen Klon des Amazon DocumentDB GitHub DocumentDB-Tools-Repos erstellen und den Anweisungen unter folgen. `README.md` Sie können das Tool von einer Amazon EC2 EC2-Instance oder einer AWS Cloud9 Umgebung aus ausführen, die in derselben Amazon VPC wie Ihr Amazon DocumentDB-Cluster läuft.

Ersetzen Sie im folgenden Beispiel jede Information durch Ihre *`user input placeholder`* eigenen Informationen.

Der folgende Code gibt Indizes aus Ihrem Amazon DocumentDB DocumentDB-Quellcluster aus:

```
python migrationtools/documentdb_index_tool.py --dump-indexes
--uri mongodb://sample-user:user-password@sample-source-cluster.node.us-
east 1.docdb.amazonaws.com:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false'
--dir ~/index.js/

2020-02-11 21:51:23,245: Successfully authenticated to database: admin2020-02-11
 21:46:50,432: Successfully connected to instance docdb-40-xx.cluster-xxxxxxxx.us-
east-1.docdb.amazonaws.com:27017
2020-02-11 21:46:50,432: Retrieving indexes from server...2020-02-11 21:46:50,440:
 Completed writing index metadata to local folder: /home/ec2-user/index.js/
```

Sobald Ihre Indizes erfolgreich exportiert wurden, stellen Sie diese Indizes in Ihrem Amazon DocumentDB DocumentDB-Zielcluster wieder her. Verwenden Sie das Amazon DocumentDB Index Tool, um die Indizes wiederherzustellen, die Sie im vorherigen Schritt exportiert haben. Der folgende Befehl stellt die Indizes in Ihrem Amazon DocumentDB DocumentDB-Zielcluster aus dem angegebenen Verzeichnis wieder her.

```
python migrationtools/documentdb_index_tool.py --restore-indexes
--uri mongodb://sample-user:user-password@sample-destination-
cluster.node.us-east 1.docdb.amazonaws.com:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false'
--dir ~/index.js/

2020-02-11 21:51:23,245: Successfully authenticated to database: admin2020-02-11
 21:51:23,245: Successfully connected to instance docdb-50-xx.cluster-xxxxxxxx.us-
east-1.docdb.amazonaws.com:27017
2020-02-11 21:51:23,264: testdb.coll: added index: _id
```

Um zu überprüfen, ob Sie die Indizes korrekt wiederhergestellt haben, stellen Sie mit der Mongo-Shell eine Verbindung zu Ihrem Amazon DocumentDB DocumentDB-Zielcluster her und listen Sie die Indizes für eine bestimmte Sammlung auf. Sehen Sie sich den folgenden Code an:

```
mongo --ssl
--host docdb-xx-xx.cluster-xxxxxxxx.us-east-1.docdb.amazonaws.com:27017
--sslCAFile rds-ca-2019-root.pem --username documentdb --password documentdb

db.coll.getIndexes()
```

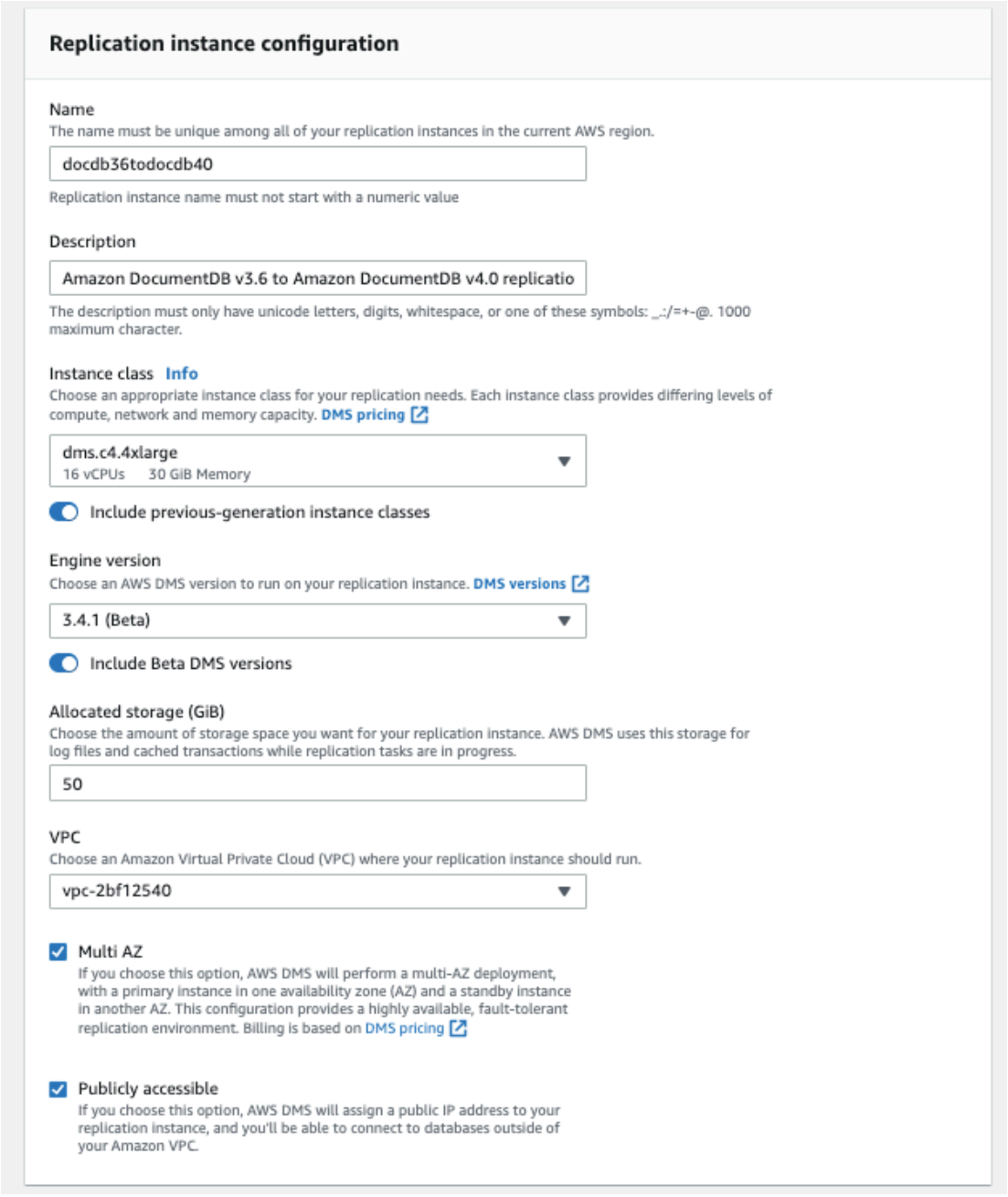
Schritt 4: Erstellen Sie ein AWS DMS replication instance

Eine AWS DMS Replikationsinstanz stellt eine Verbindung her und liest Daten aus Ihrem Amazon DocumentDB DocumentDB-Quellcluster und schreibt sie in Ihren Amazon DocumentDB DocumentDB-Zielcluster. Die AWS DMS Replikationsinstanz kann sowohl Massenlade- als auch CDC-Operationen ausführen. Der Großteil dieser Verarbeitung findet im Speicher statt. Bei großen Vorgängen kann jedoch eine gewisse Pufferung auf der Festplatte erforderlich sein. Zwischengespeicherte Transaktionen und Protokolldateien werden ebenfalls auf Festplatte geschrieben. Sobald die Daten migriert sind, streamt die Replikationsinstanz auch alle Änderungsereignisse, um sicherzustellen, dass Quelle und Ziel synchron sind.

So erstellen Sie eine AWS DMS Replikationsinstanz:

1. Öffnen Sie die AWS DMS Konsole.
2. Wählen Sie im Navigationsbereich Replication instances (Replikations-Instances) aus.
3. Wählen Sie Create replication instance (Replikations-Instance erstellen) aus und geben Sie die folgenden Informationen ein:
   - Geben Sie unter Name einen Namen Ihrer Wahl ein. Beispiel, `docdb36todocdb40`.
   - Geben Sie unter Beschreibung eine Beschreibung Ihrer Wahl ein. Für lititem eine Replikationsinstanz von Amazon DocumentDB 3.6 auf Amazon DocumentDB 4.0.
   - Wählen Sie für Instance-Klasse die Größe entsprechend Ihren Anforderungen aus.
   - Wählen Sie für Engine-Version `3.4.1.`
   - Wählen Sie für Amazon VPC die Amazon VPC aus, die Ihre Amazon DocumentDB DocumentDB-Quell- und Zielcluster beherbergt.

- Verwenden Sie für Allocation Storage (GiB) den Standardwert von 50 GiB. Wenn Sie einen Workload mit hohem Schreibdurchsatz haben, erhöhen Sie diesen Wert entsprechend Ihrer Arbeitslast.
- Wählen Sie für Ja Multi-AZ, wenn Sie Hochverfügbarkeits- und Failover-Unterstützung benötigen.
- Für Publicly accessible (Öffentlich zugänglich) aktivieren Sie diese Option.



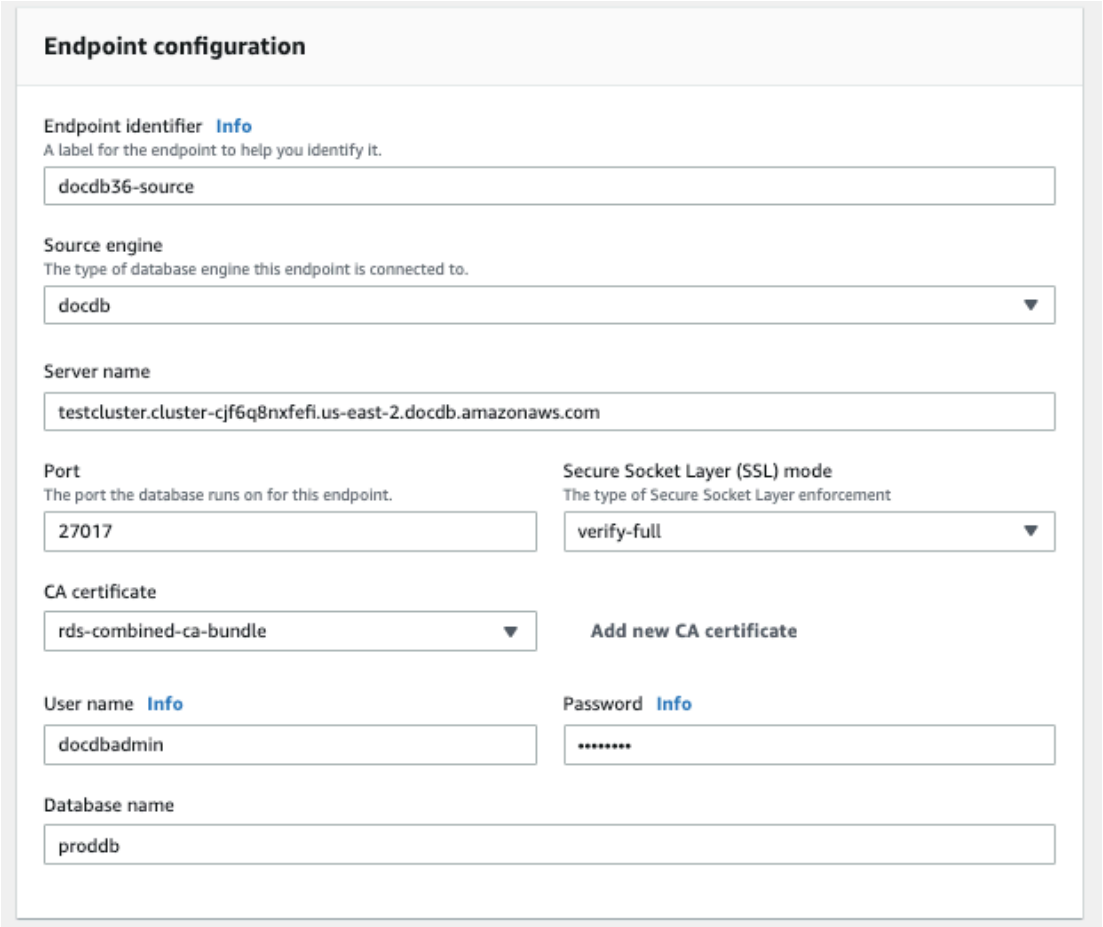
4. Wählen Sie Create replication instance (Replikations-Instance erstellen) aus.

Schritt 5: Erstellen Sie eine AWS DMS Quell-Endpunkt

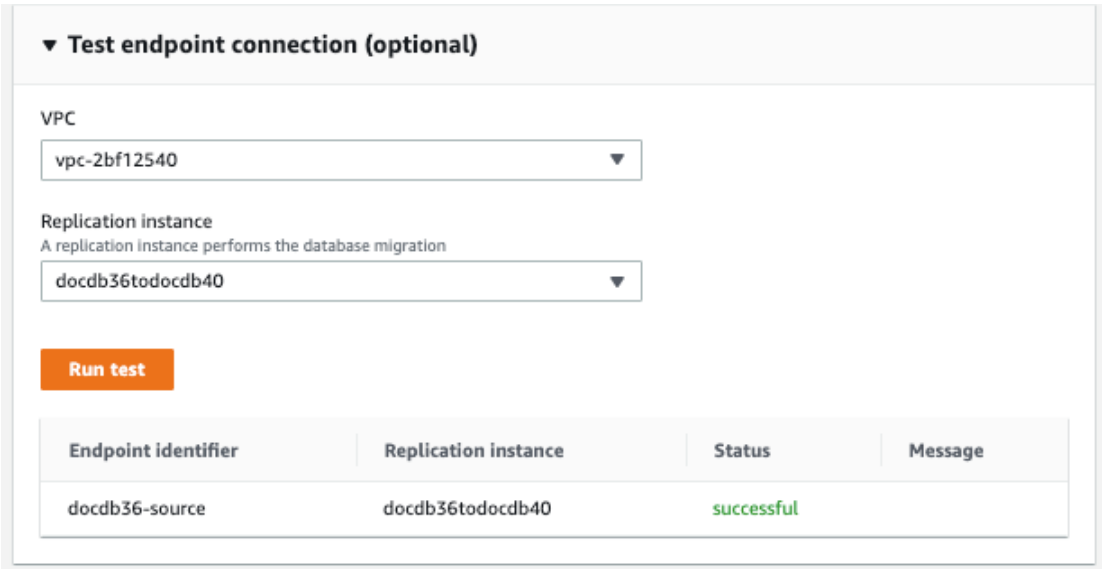
Der Quellendpunkt wird für den Amazon DocumentDB DocumentDB-Quellcluster verwendet.

Um einen Quellendpunkt zu erstellen

1. Öffnen Sie die AWS DMS Konsole.
2. Wählen Sie im Navigationsbereich Endpunkte aus.
3. Wählen Sie die folgenden Informationen aus `Create endpoint` und geben Sie sie ein:
   - Für Endpoint type (Endpunkttyp) wählen Sie Source (Quelle) aus.
   - >Geben Sie für Endpoint Identifier beispielsweise `docdb-source` einen Namen ein, den Sie sich leicht merken können.
   - Wählen Sie für Source Engine die Option`docdb`.
   - Geben Sie als Servername den DNS-Namen Ihres Amazon DocumentDB DocumentDB-Quell-Clusters ein.
   - Geben Sie für Port die Portnummer Ihres Amazon DocumentDB DocumentDB-Quell-Clusters ein.
   - Wählen Sie `verify-full` für den SSL-Modus.
   - Wählen Sie für CA-Zertifikat die Option Neues CA-Zertifikat hinzufügen aus. Laden Sie das neue CA-Zertifikat (neues CA-Zertifikat herunter, um ein TLS-Verbindungspaket zu erstellen. Geben Sie als Certificate Identifier ein `rds-combined-ca-bundle`. Wählen Sie unter Import certificate file (Zertifikatsdatei importieren) die Option Choose file (Datei auswählen) und navigieren Sie zu der `.pem`-Datei, die Sie zuvor heruntergeladen haben. Wählen Sie die Datei aus und öffnen Sie sie. Wählen Sie Zertifikat importieren und wählen Sie dann `rds-combined-ca-bundle` aus der Dropdownliste Zertifikat auswählen
   - Geben Sie unter Benutzername den primären Benutzernamen Ihres Amazon DocumentDB DocumentDB-Quell-Clusters ein.
   - Geben Sie unter Passwort das primäre Passwort Ihres Amazon DocumentDB DocumentDB-Quell-Clusters ein.
   - Geben Sie unter Datenbankname den Datenbanknamen ein, für den Sie ein Upgrade durchführen möchten.



4. Testen Sie Ihre Verbindung, um sicherzustellen, dass sie erfolgreich eingerichtet wurde.



5. Klicken Sie auf Endpunkt erstellen.

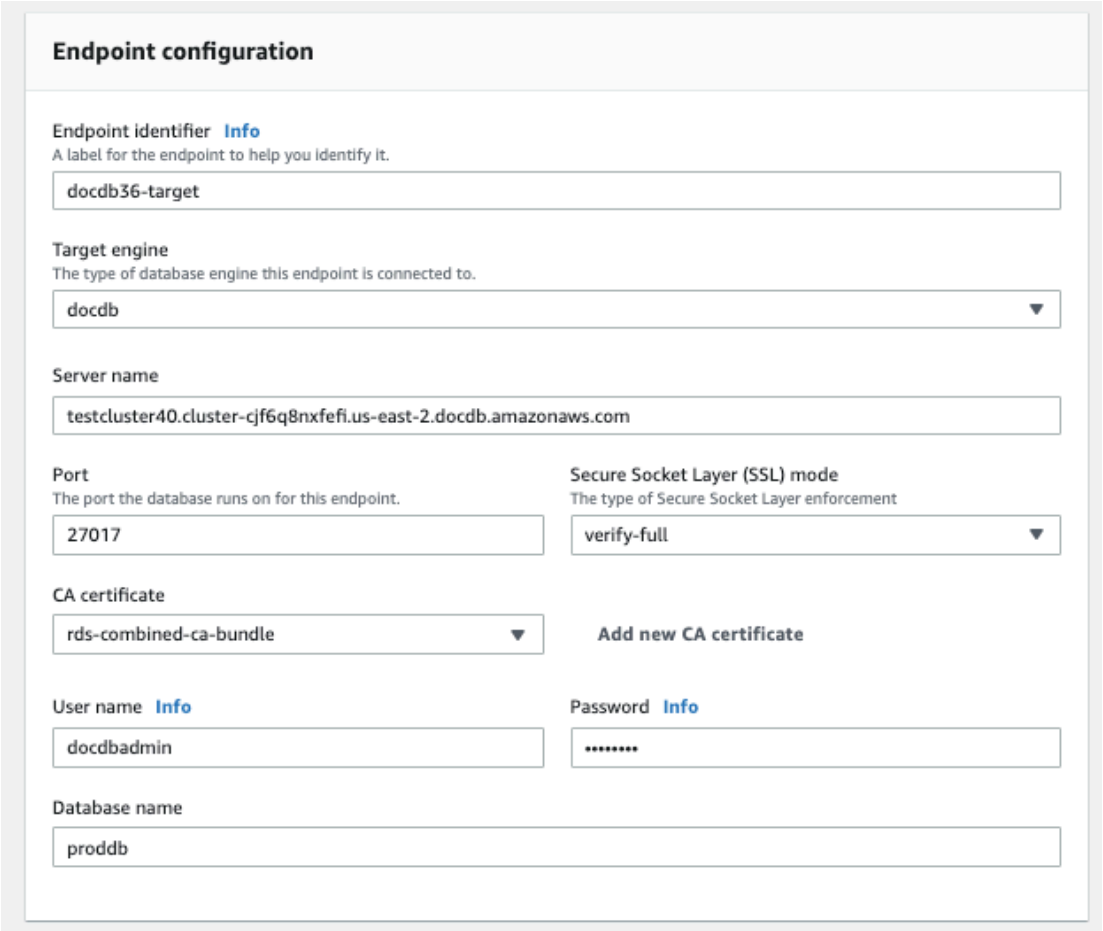
> **Note**
>
> AWS DMS kann jeweils nur eine Datenbank migrieren.

Schritt 6: Erstellen Sie eine AWS DMS Zielendpunkt

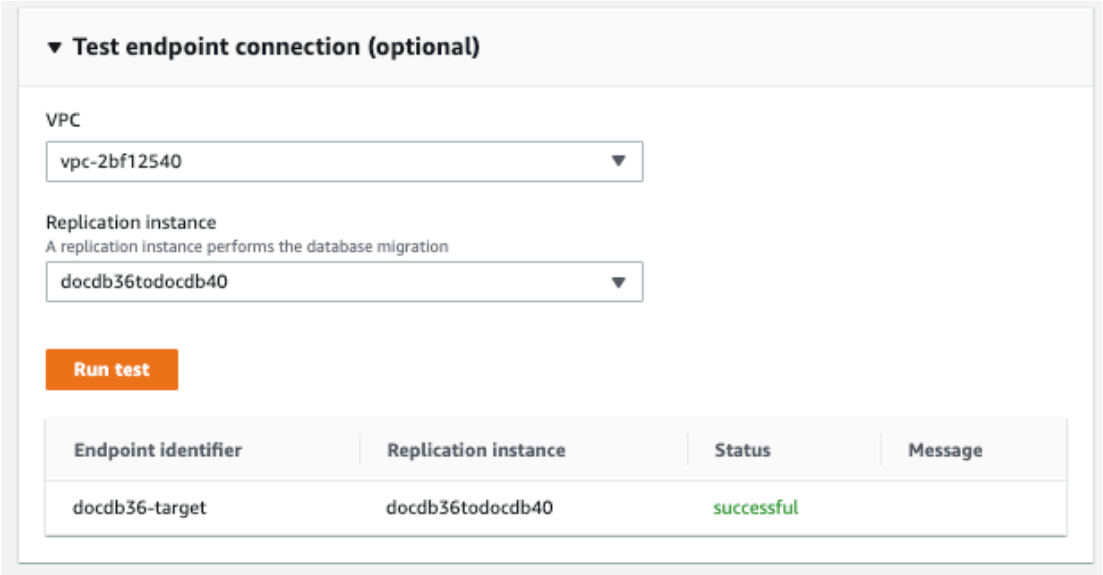
Der Zielendpunkt ist für Ihren Amazon DocumentDB DocumentDB-Zielcluster.

Um einen Zielendpunkt zu erstellen:

1. Öffnen Sie die AWS DMS -Konsole.
2. Wählen Sie im Navigationsbereich Endpunkte aus.
3. Wählen Sie Create endpoint (Endpunkt erstellen) und geben Sie die folgenden Informationen ein:
   - Für Endpoint type (Endpunkttyp) wählen Sie Target (Ziel) aus.
   - Für Endpoint Identifier (Endpunkt-ID) geben Sie einen Namen ein, der leicht zu merken ist, zum Beispiel `docdb-target`.
   - Wählen Sie für Source Engine`docdb`.
   - Geben Sie als Servername den DNS-Namen Ihres Amazon DocumentDB DocumentDB-Ziel-Clusters ein.
   - Geben Sie für Port die Portnummer Ihres Amazon DocumentDB DocumentDB-Ziel-Clusters ein.
   - Wählen Sie `verify-full` für den SSL-Modus.
   - Wählen Sie für CA-Zertifikat das vorhandene `rds-combined-ca-bundle` Zertifikat aus der Drop-down-Liste Zertifikat auswählen aus.
   - Geben Sie unter Benutzername den primären Benutzernamen Ihres Amazon DocumentDB DocumentDB-Ziel-Clusters ein.
   - Geben Sie unter Passwort das primäre Passwort Ihres Amazon DocumentDB DocumentDB-Ziel-Clusters ein.
   - Geben Sie als Datenbankname denselben Datenbanknamen ein, den Sie zur Einrichtung Ihres Quellendpunkts verwendet haben.



4. Testen Sie Ihre Verbindung, um sicherzustellen, dass sie erfolgreich eingerichtet wurde.



5. Klicken Sie auf Endpunkt erstellen.

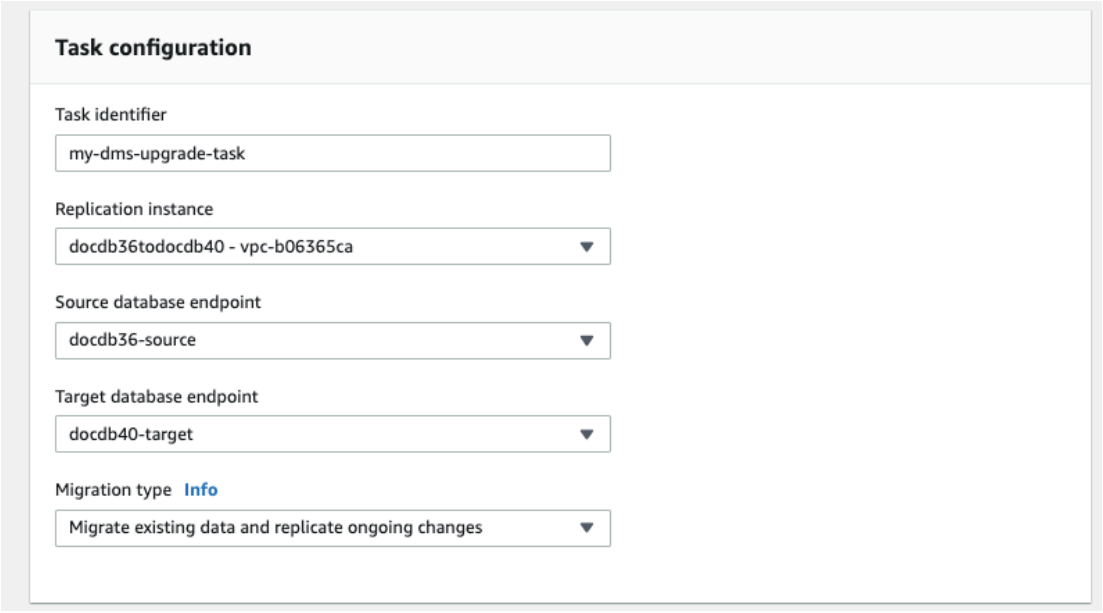
Schritt 7: Erstellen Sie eine Migrationsaufgabe und führen Sie sie aus

Eine AWS DMS Aufgabe verbindet die Replikationsinstanz mit Ihrer Quell- und Zielinstanz. Wenn Sie eine Migrationsaufgabe erstellen, geben Sie den Quellendpunkt, den Zielendpunkt, die Replikationsinstanz und alle gewünschten Migrationseinstellungen an. Eine AWS DMS Aufgabe kann mit drei verschiedenen Migrationstypen erstellt werden: Migration vorhandener Daten, Migration vorhandener Daten und Replikation laufender Änderungen oder nur Datenänderungen. Da der Zweck dieser Anleitung darin besteht, einen Amazon DocumentDB-Cluster mit minimaler Ausfallzeit zu aktualisieren, nutzen die Schritte die Option, vorhandene Daten zu migrieren und laufende Änderungen zu replizieren. Mit dieser Option werden Änderungen bei der Migration Ihrer vorhandenen Daten AWS DMS erfasst. AWS DMS erfasst weiterhin Änderungen und wendet sie an, auch nachdem die Massendaten geladen wurden. Schließlich sind die Quell- und Zieldatenbanken synchronisiert, unter Berücksichtigung einer minimalen Ausfallzeit während der Migration.

Im Folgenden finden Sie die Schritte zum Erstellen einer Migrationsaufgabe für eine Migration mit minimalen Ausfallzeiten:

1. Öffnen Sie die AWS DMS Konsole.
2. Wählen Sie im Navigationsbereich die Option Datenbankmigrationsaufgaben aus.
3. Wählen Sie Aufgabe zur Datenbankmigration erstellen und geben Sie die folgenden Informationen in den Abschnitt Aufgabenkonfiguration ein:

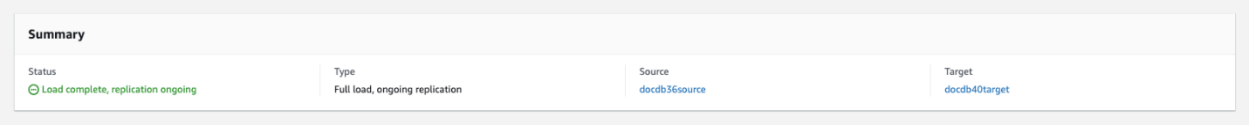
   - Geben Sie unter Task-ID beispielsweise einen Namen ein, den Sie sich leicht merken können`my-dms-upgrade-task`.
   - Geben Sie für Descriptive Amazon Resource Name (ARN) einen benutzerfreundlichen Namen ein, um den standardmäßigen DMS-ARN zu überschreiben.
   - Für Replication instance (Replikations-Instance) wählen Sie die Replikations-Instance aus, die Sie in Schritt 4: Erstellen Sie ein AWS DMS replication instance erstellt haben.
   - Wählen Sie für Quelldatenbank-Endpunkt den Quellendpunkt aus, in dem Sie ihn erstellt haben. Schritt 5: Erstellen Sie eine AWS DMS Quell-Endpunkt
   - Wählen Sie für Zieldatenbank-Endpunkt den Zielendpunkt aus, in dem Sie ihn erstellt habenSchritt 6: Erstellen Sie eine AWS DMS Zielendpunkt.
   - Wählen Sie als Migrationstyp die Option Migrieren und replizieren aus.



4. Geben Sie im Abschnitt Aufgabeneinstellungen die folgenden Informationen ein:

   - Wählen Sie für den Abschnitt Zieltabellenvorbereitungsmodus die Option Nichts tun aus. Dadurch wird sichergestellt, dass die in Schritt 3 erstellten Indizes nicht gelöscht werden.
   - Wählen Sie im Unterabschnitt Aufgabenprotokolle die Option Protokolle aktivieren aus. CloudWatch
   - Wählen Sie für die Startkonfiguration der Migrationsaufgabe die Option Automatisch bei Erstellung aus. Dadurch wird die Migrationsaufgabe automatisch gestartet, sobald Sie sie erstellt haben.
   - Wählen Sie Datenbankmigrationsaufgabe erstellen aus.

AWS DMS beginnt jetzt mit der Migration von Daten aus Ihrem Amazon DocumentDB DocumentDB-Quellcluster zu Ihrem Amazon DocumentDB DocumentDB-Zielcluster. Der Aufgabenstatus sollte sich von „Wird gestartet“ auf „Wird ausgeführt“ ändern. Sie können den Fortschritt überwachen, indem Sie in der AWS DMS Konsole Aufgaben auswählen. Nach mehreren minutes/hours Schritten (abhängig von der Größe Ihrer Migration) sollte sich der Status von „Laden abgeschlossen“ ändern, wobei die Replikation läuft. Das bedeutet, dass AWS DMS eine Volllastmigration Ihres Amazon DocumentDB-Quell-Clusters zu einem Amazon DocumentDB-Zielcluster abgeschlossen wurde und jetzt Änderungsereignisse repliziert werden.



Irgendwann werden Ihre Quelle und Ihr Ziel synchron sein. Sie können überprüfen, ob sie synchron sind, indem Sie einen `count()` Vorgang für Ihre Sammlungen ausführen, um zu überprüfen, ob alle Änderungsereignisse migriert wurden.

Schritt 8: Ändern des Anwendungsendpunkts auf den Amazon DocumentDB DocumentDB-Zielcluster

Nachdem der vollständige Ladevorgang abgeschlossen ist und der CDC-Prozess kontinuierlich repliziert wird, können Sie den Datenbankverbindungsendpunkt Ihrer Anwendung von Ihrem Amazon DocumentDB DocumentDB-Quellcluster auf Ihren Amazon DocumentDB DocumentDB-Zielcluster ändern.

Long-Term Support- (LTS, Langzeit-Support) Versionen nutzen

Amazon DocumentDB bezeichnet bestimmte Versionen als Long-Term Support (LTS) -Versionen. Datenbank-Cluster, die LTS verwenden, können länger auf derselben Version bleiben und weniger Upgrade-Zyklen durchlaufen als Cluster, die Nicht-LTS-Versionen verwenden. LTS-Versionen werden nur bei kritischen Stabilitäts- und Sicherheitspatches gepatcht, ohne dass neue Funktionen eingeführt werden. Zusätzliche optionale Patches können veröffentlicht werden, z. B. Metriken zum Debuggen kritischer Fehler. Diese werden zusammen mit den erforderlichen Patches angewendet.

Aktuelle LTS-Version

Die aktuelle LTS-Version für Amazon DocumentDB ist:

- Nebenversion 5.0.0

Informationen zur Bestimmung der Version, auf der sich Ihr Cluster befindet, finden Sie unterthe section called “Wie überprüfe ich, ob mein vorhandener Engine-Cluster mit Version 5.0.0 auf LTS läuft”.

Weitere Informationen zu Supportzeitplänen und Veröffentlichungszyklen für die LTS-Versionen finden Sie unter. Unterstützungstermine der Amazon DocumentDB DocumentDB-Engine-Version

Amazon DocumentDB wird die nächste LTS-Version über Standardkommunikationskanäle ankündigen.

> **Note**
>
> Eine LTS-Version für Amazon DocumentDB 8.0 wurde noch nicht festgelegt. Wenn Sie LTS benötigen, verwenden Sie Engine-Version 5.0.0 mit Patch-Version 3.0.17983 oder höher.

Wer sollte LTS verwenden

Amazon DocumentDB LTS-Versionen sind für Cluster mit begrenzten Upgrade-Fenstern konzipiert. LTS ist ideal für Produktionsumgebungen, in denen sich die Datenbankverfügbarkeit direkt auf den Geschäftsbetrieb auswirkt.

Ziehen Sie LTS-Versionen in Betracht, wenn:

- Ihre Anwendung hat strenge Verfügbarkeitsanforderungen, die die Upgrade-Möglichkeiten einschränken
- Sie verfügen über alle erforderlichen Funktionen und Erweiterungen der Datenbank-Engine, die Ihre Anwendung benötigt

Ziehen Sie stattdessen Standardversionen in Betracht, wenn:

- Ihre Anwendung kann minimale Unterbrechungen verkraften
- Sie möchten die neuesten Funktionen und Verbesserungen, sobald sie veröffentlicht werden

> **Important**
>
> - Upgrades sind einseitig — Sie können nach dem Upgrade kein Downgrade mehr durchführen
> - Bei Ihrem Cluster kommt es während des Upgrade-Vorgangs zu kurzen Ausfallzeiten

Wie erstelle ich einen neuen LTS-Cluster

Gehen Sie folgendermaßen vor, um einen LTS-Cluster zu erstellen:

1. Erstellen Sie in der Management Console einen neuen Engine-Cluster der Version 5.0.0
2. Stellen Sie nach der Clustererstellung sicher, dass Sie die Engine-Version 5.0.0 verwenden. Verwenden Sie dazu: AWS CLI

```
aws docdb describe-db-clusters \
    --db-cluster-identifier mydocdbcluster \
    --query 'DBClusters[0].EngineVersion'
```

Die Ausgabe dieser Operation sieht in etwa wie folgt aus:

```
"5.0.0"
```

Wie führe ich ein Upgrade von Engine-Version 3.6.0 oder 4.0.0 auf einen 5.0.0 LTS-Cluster durch

Gehen Sie folgendermaßen vor, um ein Upgrade auf einen LTS-Cluster durchzuführen:

1. Führen Sie in der Management Console ein Upgrade der Hauptversion (MVU) Ihres Engine-Clusters der Version 3.6.0 oder 4.0.0 auf 5.0.0 durch. Anweisungen zur Durchführung einer MVU finden Sie unter. Direktes Upgrade der Hauptversion von Amazon DocumentDB
2. Stellen Sie sicher, dass Sie nach dem Upgrade die Engine-Version 5.0.0 verwenden, indem Sie: AWS CLI

```
aws docdb describe-db-clusters \
    --db-cluster-identifier mydocdbcluster \
    --query 'DBClusters[0].EngineVersion'
```

Wie überprüfe ich, ob mein vorhandener Engine-Cluster mit Version 5.0.0 auf LTS läuft

So überprüfen Sie, ob auf Ihrem Engine-Cluster mit Version 5.0.0 eine LTS-Version ausgeführt wird:

1. Stellen Sie sicher, dass Sie die Engine-Version 5.0.0 verwenden, indem Sie: AWS CLI

```
aws docdb describe-db-clusters \
    --db-cluster-identifier mydocdbcluster \
```

```
    --query 'DBClusters[0].EngineVersion'
```

2. Aktualisieren Sie bei Bedarf Ihren vorhandenen Engine-Cluster mit Version 5.0.0. Anweisungen zur Durchführung eines Patch-Updates finden Sie unter Durchführung eines Patch-Updates für die Engine-Version eines Clusters.

Erweiterter Support für Amazon DocumentDB

Amazon DocumentDB Extended Support ermöglicht es Ihnen, eine Engine-Version nach Ablauf des Standard-Supportdatums gegen Aufpreis weiter zu verwenden. Ihre Cluster werden nach Ablauf des Standard-Supportdatums für den erweiterten Support registriert. Wenn Sie die Upgrades nicht bis zum Ende des Standard-Supportdatums der Engine-Version abschließen, wird Ihren Clustern der Tarif für den erweiterten Support in Rechnung gestellt.

Topics

- Überblick über den erweiterten Support
- Gebühren für erweiterten Support
- Verantwortlichkeiten

Amazon DocumentDB Extended Support bietet die folgenden Updates und technischen Support:

- Sicherheitsupdates für kritische und hohe CVEs für Ihre Instance oder Ihren Cluster
- Bugfixes und Patches für kritische Probleme
- Die Möglichkeit, Support-Anfragen zu öffnen und Hilfe bei der Fehlerbehebung im Rahmen des standardmäßigen Amazon DocumentDB-Service Level Agreements zu erhalten

Dieses kostenpflichtige Angebot gibt Ihnen mehr Zeit für ein Upgrade auf eine unterstützte Engine-Version. Beispiel: Das Ende des Standardsupports von Amazon DocumentDB für Version 3.6 ist der 30. März 2026, Sie sind jedoch nicht bereit, vor diesem Datum auf Version 5.0 oder 8.0 zu aktualisieren. In diesem Fall registriert Amazon DocumentDB Ihren Cluster der Version 3.6 am 31. März 2026 bei Extended Support, sodass Sie Amazon DocumentDB Version 3.6 weiterhin verwenden können. Ab dem 1. Juli 2026 wird Ihnen eine Extended AWS Support-Gebühr für Amazon DocumentDB-Cluster der Version 3.6 berechnet.

Amazon DocumentDB Extended Support ist bis zu 3 Jahre nach dem Standard-Supportdatum der Engine-Version verfügbar. Wenn Sie Ihre Engine-Version nach Ablauf dieser Zeit nicht auf eine

unterstützte Version aktualisiert haben, aktualisiert Amazon DocumentDB Ihre Engine-Version. Führen Sie so schnell wie möglich ein Upgrade auf eine unterstützte Engine-Version durch.

Datum des Endes des Standardsupports für Amazon DocumentDB Version 3.6 und Datum des erweiterten Supports:

- Veröffentlichung der Version 3.6 — 9. Januar 2019
- Ende der Standardunterstützung für Version 3.6 — 30. März 2026
- Beginn der Version 3.6 Extended Support, Preise für das erste Jahr — 1. Juli 2026
- Beginn der Version 3.6 Extended Support, Preise für das dritte Jahr — 31. März 2028
- Ende des erweiterten Supports von Version 3.6 — 30. März 2029

Überblick über den erweiterten Support

Amazon DocumentDB aktualisiert Ihren Cluster auf die letzte veröffentlichte Engine-Version (vor Ablauf des Extended Support-Datums), falls Ihr Cluster noch nicht über diese Version verfügt. Der Upgradezeitraum beginnt erst nach Ablauf des Standard-Supportdatums für diese Engine-Version.

Sie können neue Cluster mit einer Engine-Version erstellen, die das Ende des Standard-Supports erreicht hat, wenn Sie bereits einen Cluster für diese Engine-Version haben. Amazon DocumentDB registriert diese neuen Cluster beim Amazon DocumentDB Extended Support und berechnet Ihnen dieses Angebot in Rechnung. Wenn Sie vor dem Ende des Standardsupports von Version 3.6 auf eine Engine-Version aktualisieren, für die noch der Standardsupport von Amazon DocumentDB verfügbar ist, wird Ihnen kein Aufpreis für den erweiterten Support berechnet.

Sie können die Registrierung für Amazon DocumentDB Extended Support jederzeit beenden. Um die Registrierung zu beenden, aktualisieren Sie jeden registrierten Cluster auf eine neuere Engine-Version, für die noch immer der Amazon DocumentDB-Standardsupport gilt. Das Ende der Registrierung für Amazon DocumentDB Extended Support tritt an dem Tag in Kraft, an dem Sie ein Upgrade auf eine neuere Engine-Version abschließen, für die noch der Amazon DocumentDB-Standardsupport gilt.

Weitere Informationen zum Ende des Standard-Supports für Amazon DocumentDB und zum Ende des erweiterten Supports für Amazon DocumentDB finden Sie unter. Unterstützungstermine der Amazon DocumentDB DocumentDB-Engine-Version

Gebühren für erweiterten Support

Für alle Cluster, die den erweiterten Support nutzen, fallen ab dem 1. Juli 2026 Gebühren an. Informationen zum Ende des Standardsupports für Amazon DocumentDB Version 3.6 finden Sie unter. Unterstützungstermine der Amazon DocumentDB DocumentDB-Engine-Version

Die zusätzliche Gebühr für Amazon DocumentDB Extended Support endet, wenn Sie eine der folgenden Maßnahmen ergreifen:

- Führen Sie ein Upgrade auf eine Engine-Version durch, die unter den Standardsupport fällt
- Löschen Sie den Cluster, auf dem Version 3.6 ausgeführt wird, nach Ablauf des Standardsupports

Zum Beispiel tritt Amazon DocumentDB Version 3.6 am 31. März 2026 in den erweiterten Support ein, aber die Gebühren beginnen erst am 1. Juli 2026. Wenn Sie Ihren Amazon DocumentDB-Cluster Version 3.6 am 31. Juli 2026 auf Amazon DocumentDB-Version 5.0 aktualisieren, werden Ihnen nur 30 Tage Extended Support für Amazon DocumentDB Version 3.6 in Rechnung gestellt. Weitere Informationen finden Sie unter Amazon DocumentDB-Preise.

Verantwortlichkeiten

Im Folgenden sind die Zuständigkeiten von Amazon DocumentDB und Ihre Pflichten gegenüber Amazon DocumentDB Extended Support aufgeführt.

Verantwortlichkeiten von Amazon DocumentDB

Nach Ablauf des Standardsupports für Amazon DocumentDB Version 3.6 wendet Amazon DocumentDB Patches, Bugfixes und Upgrades für Cluster der Version 3.6 an, die für Extended Support angemeldet sind.

Ihre Verantwortlichkeiten

Sie sind dafür verantwortlich, Ihre Engine vor Ablauf des erweiterten Supports von Amazon DocumentDB Version 3.6 auf eine neuere Engine-Version zu aktualisieren. Wenn Sie Ihren Cluster nicht aktualisieren, versucht Amazon DocumentDB nach Ablauf des Extended Supports für Amazon DocumentDB Version 3.6, Ihren Cluster auf die neueste Engine-Version zu aktualisieren, die vom Amazon DocumentDB-Standardsupport unterstützt wird.

Funktionen und Konfigurationen von Amazon DocumentDB

In den folgenden Tabellen werden die Funktionen, Funktionen und unterstützten Instance-Typen der Amazon DocumentDB-Engine-Versionen und Elastic Clusters verglichen. Verwenden Sie diese Tabellen, um zu ermitteln, welche Engine-Version Ihren Workload-Anforderungen am besten entspricht.

Status der Engine

Motorstatus nach Version

Funktion/Fähigkeit	v3.6	v4.0	v5.0	v8.0	
Status der Unterstützung	EOL: März 2026	Aktiv	Aktiv (LTS verfügbar)	Aktiv (GA Nov. 2025)	
Erweiterter Support verfügbar	Ja (bezahlt, bis März 2029)	N/A	N/A	N/A	
LTS-Version verfügbar	Nein	Nein	Ja (v5.0 LTS)	Nein	
Elastische Cluster	Nein	Nein	Ja	Nein	

Cluster-Typen

Clustertypen nach Version

Cluster Type (Clustertyp	v3.6	v4.0	v5.0	v8.0	Elastische Cluster
Instance-based Cluster	Ja	Ja	Ja	Ja	Ja
Serverlose Instanzen	Nein	Nein	Ja	Ja	Nein

Features

Funktionen und Konfigurationen nach Engine-Version

Feature	v3.6	v4.0	v5.0	v8.0	Elastische Cluster
Globale Cluster	Nein	Ja	Ja	Ja	Nein
Multi-AZ (3 AZs)	Ja	Ja	Ja	Ja	Ja (pro Shard)
Read Replicas werden unterstützt	15	15	15	15	15 pro Shard
ACID-Transaktionen	Nein	Ja	Ja	Ja	Nein
Wiederholbare Schreibvorgänge	Nein	Nein	Nein	Ja (8.0.2+)	Nein
Change streams	Ja	Ja	Ja	Ja	Nein
Vektor-Suche	Nein	Nein	Ja	Ja	Nein
Performance Insights	Ja	Ja	Ja	Ja	Nein
In-place Hauptversions-Upgrade (MVU)	Ja (auf 5.0 oder 8.0)	Ja (auf 5,0 oder 8,0)	Ja (bis 8,0)	Nein (nur Ziel)	Nein
Client-side Verschlüsselung auf Feldebene (FLE)	Nein	Nein	Ja	Ja	Nein
I/O-Optimized Speicher	Nein	Nein	Ja	Ja	Nein
Rotation des TLS-Zertifikats (kein Neustart)	Ja (Patch 1.0.208662+)	Ja (Patch 2.0.10179+)	Ja (Patch 3.0.4780+)	Ja	Ja

Feature	v3.6	v4.0	v5.0	v8.0	Elastische Cluster
Cross-Region Snapshot-Kopie	Ja	Ja	Ja	Ja	Ja
Cross-account Teilen von Snapshots	Ja	Ja	Ja	Ja	Nein
Protokolle für Prüfungen und langsame Abfragen	Ja	Ja	Ja	Ja	Ja
Dual-stack IPv4/IPv6	Nein	Ja (Patch 2.0.11747 +)	Ja (Patch 3.0.15902 +)	Ja	N/A
Kollation	Nein	Nein	Nein	Ja	Nein
Ansichten	Nein	Nein	Nein	Ja	Nein
Komprimierungsunterstützung	Nein	Nein	LZ4	LZ4/Ztsd	Nein
Textindex	Nein	Nein	V1	V2	Nein
FedRampe () GovCloud	Nein	Nein	Ja	Ja	Nein

Instance-Typen

Unterstützung für Instanztypen nach Version

Instance-Typ	v3.6	v4.0	v5.0	v8.0	
db.r4 (vorherige Generation)	Ja	Nein	Nein	Nein	
db.r5	Ja	Ja	Ja	Ja	
db.r6g (Graviton 2)	Nein	Ja	Ja	Ja	

Instance-Typ	v3.6	v4.0	v5.0	v8.0	
db.r6gd (NVMe-backed Graviton 2)	Nein	Nein	Ja	Ja	
db.r8g (Graviton 4)	Nein	Nein	Ja	Ja	
db.t3.medium	Ja	Ja	Ja	Ja	
db.t4g.medium (Graviton 2)	Nein	Ja	Ja	Ja	
Serverless	Nein	Nein	Ja	Ja	

AWS Integrationen

AWS Integrationsunterstützung nach Version

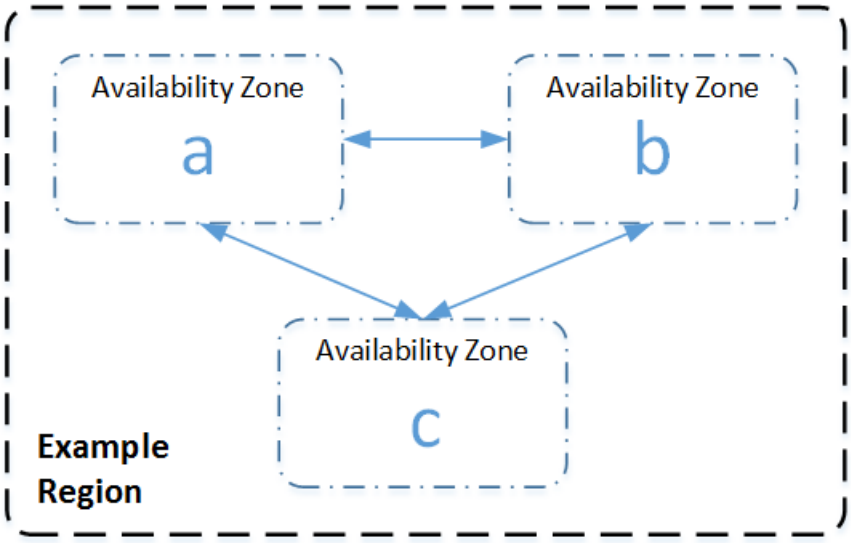
AWS Integration/Funktion	v3.6	v4.0	v5.0	v8.0	Elastische Cluster
AWS Backup	Ja	Ja	Ja	Ja	Nein
AWS Secrets Manager	Ja	Ja	Ja	Ja	Nein
AWS Lambda (natives ESM)	Nein	Ja	Ja	Nein	Nein
Amazon OpenSearch Service Ingestion () Zero-ETL	Nein	Ja	Ja	Ja	Nein
AWS DMS Unterstützung für die Migration	Ja	Ja	Ja	Ja (ab 5.0 über AWS DMS)	Ja
AWS Identity and Access Management authentication	Nein	Nein	Ja (instanzbasiert auf	Ja	Nein

AWS Integration/Funktion	v3.6	v4.0	v5.0	v8.0	Elastische Cluster
			Version 5.0)		
AWS CloudShell	Nein	Nein	Ja	Ja	Nein
AWS Key Management Service	Ja	Ja	Ja	Ja	Ja
AWS CloudTrail	Ja	Ja	Ja	Ja	Ja

Auswählen von Regionen und Availability Zones

Amazon Cloud Computing-Ressourcen werden an mehreren Standorten weltweit gehostet. Diese Standorte bestehen aus AWS-Regionen Availability Zones. Jeder AWS-Regionist ein separates geografisches Gebiet. Jede Region verfügt über mehrere isolierte Standorte, die als Availability Zones bezeichnet werden. Amazon DocumentDB bietet Ihnen die Möglichkeit, Ressourcen wie Instances und Daten an mehreren Standorten zu platzieren. Ressourcen werden nur repliziert, AWS-Regionen wenn Sie dies ausdrücklich tun.

Amazon betreibt hochmoderne, hoch verfügbare Rechenzentren. In seltenen Fällen kann es aber zu Ausfällen kommen, die die Verfügbarkeit von Instances desselben Standorts beeinträchtigen. Wenn Sie Ihre gesamten Instances an einem einzelnen Standort hosten, der von einem Ausfall dieser Art betroffen ist, ist keine Ihrer Instances verfügbar. Das folgende Diagramm zeigt eine AWS-Region mit drei Availability Zones.



Beachten Sie, dass alle Regionen voneinander unabhängig sind. Jede Amazon DocumentDB DocumentDB-Aktivität, die Sie initiieren (z. B. Instances erstellen oder verfügbare Instances auflisten), wird nur in Ihrer aktuellen Standardeinstellung AWS-Region ausgeführt. Sie können die Standardregion in der Konsole ändern, indem Sie die Umgebungsvariable `EC2_REGION` festlegen. Oder Sie können sie überschreiben, indem Sie den Parameter `--region` in der AWS CLI verwenden. Weitere Informationen finden Sie unter Konfiguration der AWS Command Line Interface Abschnitte zu Umgebungsvariablen und Befehlszeilenoptionen.

Wenn Sie mit der Amazon DocumentDB-Konsole einen Cluster erstellen und sich dafür entscheiden, ein Replikat in einer anderen Availability Zone zu erstellen, erstellt Amazon DocumentDB zwei Instances. Es erstellt die primäre Instance in einer Availability Zone und die Replikat-Instance in einer anderen Availability Zone. Das Cluster-Volumen wird immer über drei Availability Zones repliziert.

Um eine Amazon DocumentDB DocumentDB-Instance in einer bestimmten Instanz zu erstellen oder mit ihr zu arbeiten AWS-Region, verwenden Sie den entsprechenden regionalen Service-Endpunkt.

Verfügbarkeit in Regionen

Amazon DocumentDB ist in den folgenden AWS Regionen verfügbar.

Von Amazon DocumentDB unterstützte Regionen

Name der Region	Region	Verfügbarkeitszonen (Datenverarbeitung)
USA Ost (Ohio)	`us-east-2`	3

Name der Region	Region	Verfügbarkeitszonen (Datenverarbeitung)
USA Ost (Nord-Virginia)	us-east-1	6
USA West (Oregon)	us-west-2	4
Afrika (Kapstadt)	af-south-1	3
Südamerika (São Paulo)	sa-east-1	3
Asien-Pazifik (Hongkong)	ap-east-1	3
Asien-Pazifik (Hyderabad)	ap-south-2	3
Asien-Pazifik (Malaysia)	ap-southeast-5	3
Asien-Pazifik (Mumbai)	ap-south-1	3
Asien-Pazifik (Osaka)	ap-northeast-3	3
Asien-Pazifik (Seoul)	ap-northeast-2	4
Asien-Pazifik (Singapur)	ap-southeast-1	3
Asien-Pazifik (Sydney)	ap-southeast-2	3
Asien-Pazifik (Jakarta)	ap-southeast-3	3
Asien-Pazifik (Melbourne)	ap-southeast-4	3

Name der Region	Region	Verfügbarkeitszonen (Datenverarbeitung)
Asien-Pazifik (Thailand)	ap-southeast-7	3
Asien-Pazifik (Tokio)	ap-northeast-1	3
Kanada (Zentral)	ca-central-1	3
Kanada West (Calgary)	ca-west-1	3
Region China (Peking)	cn-north-1	3
China (Ningxia)	cn-northwest-1	3
Europa (Frankfurt)	eu-central-1	3
Europa (Zürich)	eu-central-2	3
Europa (Irland)	eu-west-1	3
Europa (London)	eu-west-2	3
Europa (Milan)	eu-south-1	3
Europa (Paris)	eu-west-3	3
Europa (Spain)	eu-south-2	3
Europa (Stockholm)	eu-north-1	3
Mexiko (Zentral)	mx-central-1	3
Naher Osten (VAE)	me-central-1	3
Israel (Tel Aviv)	il-central-1	3
AWS GovCloud (US-West)	us-gov-west-1	3

Name der Region	Region	Verfügbarkeitszonen (Datenverarbeitung)
AWS GovCloud (US-East)	`us-gov-east-1`	3

Standardmäßig ist die Zeitzone für einen Amazon DocumentDB-Cluster Universal Time Coordinated (UTC).

Informationen zum Auffinden der Verbindungsendpunkte für Cluster und Instances in einer bestimmten Region finden Sie unter. Grundlegendes zu Amazon DocumentDB DocumentDB-Endpunkten

Zero-ETL Integration mit Amazon OpenSearch Service

Topics

- Amazon OpenSearch Service als Ziel
- Einschränkungen

Amazon OpenSearch Service als Ziel

OpenSearch Die Serviceintegration mit Amazon DocumentDB ermöglicht es Ihnen, Volllast- und Änderungsdatenereignisse in OpenSearch Domänen zu streamen. Die Aufnahmeinfrastruktur wird als OpenSearch Aufnahme-Pipelines gehostet und bietet einen hochskalierbaren Mechanismus mit niedriger Latenz, um kontinuierlich Daten aus Amazon DocumentDB-Sammlungen zu streamen.

Bei Volllast extrahiert die Zero-ETL-Integration zunächst historische Volllastdaten, um sie mithilfe einer Ingestion-Pipeline zu verwenden. OpenSearch Sobald die Volllastdaten aufgenommen sind, beginnen die OpenSearch Erfassungs-Pipelines mit dem Lesen von Daten aus Amazon DocumentDB-Change-Streams und holen schließlich auf, um die Datenkonsistenz zwischen Amazon DocumentDB und nahezu in Echtzeit aufrechtzuerhalten. OpenSearch OpenSearch speichert Dokumente in Indizes. Eingehende Daten aus einer Amazon DocumentDB-Sammlung können entweder an einen Index gesendet oder in verschiedene Indizes aufgeteilt werden. Ingestion-Pipelines synchronisieren alle Erstellungs-, Aktualisierungs- und Löschereignisse in einer Amazon DocumentDB-Sammlung entsprechend dem entsprechenden Erstellen, Aktualisieren und Löschen von OpenSearch Dokumenten, um beide Datensysteme synchron zu halten. Ingestion-Pipelines

können so konfiguriert werden, dass sie Daten aus einer Sammlung lesen und in einen Index schreiben oder Daten aus einer Sammlung lesen und bedingt an mehrere Indizes weiterleiten.

Ingestion-Pipelines können so konfiguriert werden, dass sie Daten von Amazon DocumentDB zu Amazon Service streamen. Verwenden Sie dazu: OpenSearch

- Nur Volllast
- Streamen Sie Change-Stream-Ereignisse aus Amazon DocumentDB ohne Volllast
- Volllast, gefolgt von Change-Streams aus Amazon DocumentDB

Gehen Sie wie folgt vor, um Ihre Ingestion-Pipeline einzurichten:

Schritt 1: Erstellen Sie eine Amazon OpenSearch Service-Domain oder OpenSearch eine serverlose Sammlung

Eine Amazon OpenSearch Service-Sammlung mit entsprechenden Berechtigungen zum Lesen von Daten ist erforderlich. Informationen zum Erstellen einer Sammlung finden Sie unter Erste Schritte mit Amazon OpenSearch Service oder Erste Schritte mit Amazon OpenSearch Serverless im Amazon OpenSearch Service Developer Guide. Informationen zum Erstellen einer OpenSearch AIM-Rolle mit den richtigen Berechtigungen für den Zugriff auf Schreibdaten in die Sammlung oder Domain finden Sie unter Amazon Ingestion im Amazon OpenSearch Service-Entwicklerhandbuch.

Schritt 2: Aktivieren Sie Change Streams auf dem Amazon DocumentDB-Cluster

Stellen Sie sicher, dass Change Streams für die erforderlichen Sammlungen im Amazon DocumentDB-Cluster aktiviert sind. Weitere Informationen finden Sie unter Change Streams mit Amazon DocumentDB verwenden.

Schritt 3: Richten Sie die Pipeline-Rolle mit Schreibberechtigungen in den Amazon S3-Bucket und die Zieldomäne oder Sammlung ein

Nachdem Sie Ihre Amazon DocumentDB-Sammlung erstellt und Change Stream aktiviert haben, richten Sie die Pipeline-Rolle ein, die Sie in Ihrer Pipeline-Konfiguration verwenden möchten, und fügen Sie der Rolle die folgenden Berechtigungen hinzu:

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "allowReadAndWriteToS3ForExport",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:AbortMultipartUpload",
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket/export/*"
            ]
        }
    ]
}
```

Damit eine OpenSearch Pipeline Daten in eine Domain schreiben kann, muss die OpenSearch Domain über eine Zugriffsrichtlinie auf Domänenebene verfügen, die es der Pipeline-Rolle sts_role_arn ermöglicht, darauf zuzugreifen. Die folgende Beispielrichtlinie für den Domänenzugriff ermöglicht es der benannten Pipeline-Rolle`pipeline-role`, die Sie im vorherigen Schritt erstellt haben, Daten in die angegebene Domäne zu schreiben: `ingestion-domain`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::{your-account-id}:role/{pipeline-role}"
      },
      "Action": ["es:DescribeDomain", "es:ESHttp*"],
      "Resource": "arn:aws:es:{region}:{your-account-id}:domain/{domain-name}/*"
    }
  ]
}
```

Schritt 4: Fügen Sie die Berechtigungen hinzu, die für die Erstellung der Pipeline-Rolle erforderlich sind X-ENI

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AttachNetworkInterface",
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DetachNetworkInterface",
                "ec2:DescribeNetworkInterfaces"
            ],
            "Resource": [
                "arn:aws:ec2:*:420497401461:network-interface/*",
                "arn:aws:ec2:*:420497401461:subnet/*",
                "arn:aws:ec2:*:420497401461:security-group/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:Describe*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [ "ec2:CreateTags" ],
            "Resource": "arn:aws:ec2:*:*:network-interface/*",
            "Condition": {
               "StringEquals": { "aws:RequestTag/OSISManaged": "true" }
            }
        }
    ]
```

```
}
```

Schritt 5: Erstellen Sie die Pipeline

Konfigurieren Sie eine OpenSearch Ingestion-Pipeline mit Amazon DocumentDB als Quelle. Bei dieser Beispiel-Pipeline-Konfiguration wird von der Verwendung eines Mechanismus zum Abrufen von Change-Streams ausgegangen. Weitere Informationen finden Sie unter Verwenden einer OpenSearch Ingestion-Pipeline mit Amazon DocumentDB im Amazon OpenSearch Service Developer Guide.

Einschränkungen

Die folgenden Einschränkungen gelten für die Amazon DocumentDB-Integration: OpenSearch

- Pro Pipeline wird nur eine Amazon DocumentDB-Sammlung als Quelle unterstützt.
- Cross-region Die Datenaufnahme wird nicht unterstützt. Ihr Amazon DocumentDB-Cluster und Ihre OpenSearch Domain müssen sich in derselben befinden. AWS-Region
- Cross-account Die Datenaufnahme wird nicht unterstützt. Ihr Amazon DocumentDB-Cluster und Ihre OpenSearch Ingestion-Pipeline müssen sich im selben Konto befinden. AWS
- Elastische Amazon DocumentDB-Cluster werden nicht unterstützt. Nur auf Amazon DocumentDB-Instanzen basierende Cluster werden unterstützt.
- Stellen Sie sicher, dass für den Amazon DocumentDB-Cluster die Authentifizierung mithilfe von Geheimnissen aktiviert ist. AWS AWS Geheimnisse sind der einzige unterstützte Authentifizierungsmechanismus.
- Die bestehende Pipeline-Konfiguration kann nicht aktualisiert werden, um Daten aus einer anderen Datenbank oder and/or einer anderen Sammlung aufzunehmen. Um den Namen der and/or Datenbanksammlung einer Pipeline zu aktualisieren, müssen Sie eine neue Pipeline erstellen.

Entwickeln mit Amazon DocumentDB

In diesem Abschnitt wird die Verbindung zu Ihrem Amazon DocumentDB-Cluster, das Abfragen und Verwalten von Daten, das Arbeiten mit Indizes und die Verwendung von Funktionen wie Change Streams, Transaktionen und Vektorsuche für generative KI-Anwendungen behandelt.

Themen

- Connect zu Amazon DocumentDB her
- Programm mit Amazon DocumentDB
- Abfragen in Amazon DocumentDB
- Indizes
- Change Streams mit Amazon DocumentDB verwenden
- Transaktionen in Amazon DocumentDB
- Wiederholbare Schreibvorgänge in Amazon DocumentDB
- Verwenden der Sortierung in Amazon DocumentDB 8.0
- Verwenden von Ansichten in Amazon DocumentDB 8.0
- Generative künstliche Intelligenz von Amazon DocumentDB
- Verwenden AWS Lambda mit Change-Streams

Connect zu Amazon DocumentDB her

Amazon DocumentDB bietet mehrere Verbindungsoptionen, darunter programmatische Verbindungen mithilfe von MongoDB-Treibern, Verbindungen von außerhalb einer VPC über SSH-Tunneling oder AWS VPN sowie beliebte Tools wie Studio 3T und. DataGrip Der Service unterstützt Verbindungen mit Replikatgruppen für hohe Verfügbarkeit. Für Analysen und Berichte können Sie mithilfe von JDBC- oder ODBC-Treibern eine Verbindung herstellen. EC2-Instances in derselben VPC können sich direkt mit DocumentDB-Clustern verbinden. Alle Verbindungen erfordern TLS und sind ordnungsgemäß. AWS credentials/authentication

Themen

- Programmgesteuertes Herstellen einer Verbindung zu Amazon DocumentDB
- Verbindung zu einem Amazon DocumentDB-Cluster von außerhalb einer Amazon VPC herstellen
- Verbindung zu Amazon DocumentDB als Replikatsatz herstellen

- Von Studio 3T aus eine Verbindung zu einem Amazon DocumentDB-Cluster herstellen
- Connect zu Amazon DocumentDB her mit DataGrip
- Stellen Sie mithilfe von Amazon EC2 eine Verbindung her
- Stellen Sie mithilfe des Amazon DocumentDB-JDBC-Treibers eine Verbindung her
- Stellen Sie mithilfe des Amazon DocumentDB-ODBC-Treibers eine Verbindung her

Programmgesteuertes Herstellen einer Verbindung zu Amazon DocumentDB

Dieser Abschnitt enthält Codebeispiele, die zeigen, wie Sie mithilfe verschiedener Sprachen eine Verbindung zu Amazon DocumentDB (mit MongoDB-Kompatibilität) herstellen. Die Beispiele sind in zwei Abschnitte unterteilt: Verbindung mit einem Cluster mit und ohne aktiviertem TLS (Transport Layer Security). Standardmäßig ist TLS auf Amazon DocumentDB-Clustern aktiviert. Sie können TLS jedoch bei Bedarf deaktivieren. Weitere Informationen finden Sie unter Verschlüsseln von Daten während der Übertragung.

Wenn Sie versuchen, von außerhalb der VPC, in der sich Ihr Cluster befindet, eine Verbindung zu Ihrer Amazon DocumentDB herzustellen, finden Sie weitere Informationen unter. Verbindung zu einem Amazon DocumentDB-Cluster von außerhalb einer Amazon VPC herstellen

Bevor Sie sich mit Ihrem Cluster verbinden, müssen Sie wissen, ob TLS auf dem Cluster aktiviert ist. Der nächste Abschnitt zeigt Ihnen, wie Sie den Wert des Parameters `tls` Ihres Clusters mit der AWS-Managementkonsole oder der AWS CLI bestimmen können. Danach können Sie fortfahren, indem Sie das entsprechende Codebeispiel anwenden.

Themen

- Ermitteln Sie den Wert Ihres tls-Parameters
- Verbindung bei aktiviertem TLS herstellen
- Verbindung bei deaktiviertem TLS herstellen
- Netzwerkkomprimierung in Amazon DocumentDB

Ermitteln Sie den Wert Ihres tls-Parameters

Das Ermitteln, ob in Ihrem Cluster TLS aktiviert ist, erfolgt in zwei Schritten, den Sie entweder mit AWS-Managementkonsole oder AWS CLI ausführen können.

1. Bestimmen Sie, welche Parametergruppe Ihren Cluster steuert.

   Using the AWS-Managementkonsole

   1. Melden Sie sich bei und öffnen Sie die Amazon DocumentDB-Konsole unter. AWS-Managementkonsole https://console.aws.amazon.com/docdb
   2. Wählen Sie im linken Navigationsbereich Cluster aus.
   3. Wählen Sie in der Liste der Cluster den Namen des Clusters aus.
   4. Auf der anschließend angezeigten Seite werden die Details des von Ihnen ausgewählten Clusters angezeigt. Wählen Sie die Registerkarte Configuration aus. Suchen Sie im Abschnitt Konfigurationen und Status den Namen der Parametergruppe unter Cluster-Parametergruppe.

   Using the AWS CLI

   Der folgende AWS CLI Code bestimmt, welcher Parameter Ihren Cluster steuert. Ersetzen Sie *sample-cluster* durch den Namen Ihres Clusters.

   ```
   aws docdb describe-db-clusters \
       --db-cluster-identifier sample-cluster \
       --query 'DBClusters[*].[DBClusterIdentifier,DBClusterParameterGroup]'
   ```

   Die Ausgabe dieser Operation sieht in etwa wie folgt aus:

   ```
   [
          [
              "sample-cluster",
              "sample-parameter-group"
          ]
   ]
   ```

2. Bestimmen Sie den Wert des Parameters **tls** in der Parametergruppe Ihres Clusters.

   Using the AWS-Managementkonsole

   1. Wählen Sie im Navigationsbereich Parameter groups (Parametergruppen) aus.
   2. Wählen Sie im Fenster Cluster-Parametergruppen den Namen Ihrer Cluster-Parametergruppe aus Schritt 1d aus.

3. Auf der Ergebnisseite werden die Parameter der Cluster-Parametergruppe angezeigt. Hier können Sie den Wert des `tls`-Parameters sehen. Informationen zum Ändern dieses Parameters finden Sie unter Ändern der Amazon DocumentDB-Cluster-Parametergruppen.

Using the AWS CLI

Sie können den `describe-db-cluster-parameters` AWS CLI Befehl verwenden, um die Details der Parameter in Ihrer Cluster-Parametergruppe anzuzeigen.

- **`--describe-db-cluster-parameters`**— Um alle Parameter innerhalb einer Parametergruppe und ihrer Werte aufzulisten.
  - **`--db-cluster-parameter-group name`** – Erforderlich. Der Name Ihrer Cluster-Parametergruppe.

Ersetzen Sie in den folgenden Beispielen jedes *`user input placeholder`* durch die Informationen Ihres Clusters.

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

Die Ausgabe dieser Operation sieht in etwa wie folgt aus:

```
{
        "Parameters": [
            {
                "ParameterName": "profiler_threshold_ms",
                "ParameterValue": "100",
                "Description": "Operations longer than profiler_threshold_ms
 will be logged",
                "Source": "system",
                "ApplyType": "dynamic",
                "DataType": "integer",
                "AllowedValues": "50-2147483646",
                "IsModifiable": true,
                "ApplyMethod": "pending-reboot"
            },
            {
                "ParameterName": "tls",
```

```
                "ParameterValue": "disabled",
                "Description": "Config to enable/disable TLS",
                "Source": "user",
                "ApplyType": "static",
                "DataType": "string",
                "AllowedValues": "disabled,enabled,fips-140-3",
                "IsModifiable": true,
                "ApplyMethod": "pending-reboot"
            }
        ]
}
```

> **Note**
>
> Amazon DocumentDB unterstützt FIPS 140-3-Endpunkte ab Amazon DocumentDB 5.0-Clustern (Engine-Version 3.0.3727) in den folgenden Regionen: ca-central-1, us-west-2, us-east-1, us-east-2, us-gov-east-1, us-gov-west-1.

Nachdem Sie den Wert des `tls`-Parameters bestimmt haben, fahren Sie mit dem Herstellen einer Verbindung zu Ihrem Cluster fort, indem Sie eines der Codebeispiele in den folgenden Abschnitten verwenden.

- Verbindung bei aktiviertem TLS herstellen
- Verbindung bei deaktiviertem TLS herstellen

Verbindung bei aktiviertem TLS herstellen

Um ein Codebeispiel für die programmgesteuerte Verbindung zu einem Amazon DocumentDB-Cluster anzuzeigen, wählen Sie die entsprechende Registerkarte für die Sprache, die Sie verwenden möchten. TLS-enabled

Um Daten während der Übertragung zu verschlüsseln, laden Sie den öffentlichen Schlüssel für Amazon DocumentDB herunter, der mit der folgenden Operation benannt wurde `global-bundle.pem`.

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

Wenn sich Ihre Anwendung auf Microsoft Windows befindet und eine PKCS7-Datei benötigt, können Sie das PKCS7-Zertifikat-Bundle herunterladen. Dieses Paket enthält sowohl das Zwischen- als auch das Stammzertifikat unter. https://truststore.pki.rds.amazonaws.com/global/global-bundle.p7b

Python

Der folgende Code zeigt, wie Sie mithilfe von Python eine Verbindung zu Amazon DocumentDB herstellen, wenn TLS aktiviert ist.

Ersetzen Sie im folgenden Beispiel jedes *user input placeholder* durch die Informationen Ihres Clusters.

```
import pymongo
import sys

##Create a MongoDB client, open a connection to Amazon DocumentDB as a replica set
 and specify the read preference as secondary preferred
client = pymongo.MongoClient('mongodb://sample-user:password@sample-
cluster.node.us-east-1.docdb.amazonaws.com:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

##Specify the database to be used
db = client.sample_database

##Specify the collection to be used
col = db.sample_collection

##Insert a single document
col.insert_one({'hello':'Amazon DocumentDB'})

##Find the document that was previously written
x = col.find_one({'hello':'Amazon DocumentDB'})

##Print the result to the screen
print(x)

##Close the connection
client.close()
```

Node.js

Der folgende Code zeigt, wie Sie eine Verbindung zu Amazon DocumentDB herstellen Node.js , wenn TLS aktiviert ist.

> **Important**
>
> Es gibt eine bekannte Einschränkung bei Node.js Treibern, die älter als Version 6.13.1 sind und die derzeit von der IAM-Identitätsauthentifizierung für Amazon DocumentDB nicht unterstützt werden. Node.js Treiber und Tools, die Node.js Treiber verwenden (z. B. Mongosh), müssen aktualisiert werden, um die Node.js Treiberversion 6.13.1 oder höher zu verwenden.

Ersetzen Sie im folgenden Beispiel jedes durch die Informationen *user input placeholder* Ihres Clusters.

```
var MongoClient = require('mongodb').MongoClient

//Create a MongoDB client, open a connection to DocDB; as a replica set,
//  and specify the read preference as secondary preferred

var client = MongoClient.connect(
'mongodb://sample-user:password@sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017/sample-database?
tls=true&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false',
{
  tlsCAFile: `global-bundle.pem` //Specify the DocDB; cert
},
function(err, client) {
    if(err)
        throw err;

    //Specify the database to be used
    db = client.db('sample-database');

    //Specify the collection to be used
    col = db.collection('sample-collection');

    //Insert a single document
    col.insertOne({'hello':'Amazon DocumentDB'}, function(err, result){
      //Find the document that was previously written
      col.findOne({'hello':'Amazon DocumentDB'}, function(err, result){
        //Print the result to the screen
        console.log(result);

        //Close the connection
```

```
        client.close()
      });
   });
});
```

PHP

Der folgende Code zeigt, wie Sie mithilfe von PHP eine Verbindung zu Amazon DocumentDB herstellen, wenn TLS aktiviert ist.

Ersetzen Sie im folgenden Beispiel jedes *user input placeholder* durch die Informationen Ihres Clusters.

```
<?php
//Include Composer's autoloader
require 'vendor/autoload.php';

$TLS_DIR = "/home/ubuntu/global-bundle.pem";

//Create a MongoDB client and open connection to Amazon DocumentDB
$client = new MongoDB\Client("mongodb://sample-user:password@sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017/?retryWrites=false", ["tls" => "true", "tlsCAFile"
 => $TLS_DIR ]);

//Specify the database and collection to be used
$col = $client->sampledatabase->samplecollection;

//Insert a single document
$result = $col->insertOne( [ 'hello' => 'Amazon DocumentDB'] );

//Find the document that was previously written
$result = $col->findOne(array('hello' => 'Amazon DocumentDB'));

//Print the result to the screen
print_r($result);
?>
```

Go

Der folgende Code zeigt, wie Sie mithilfe von Go eine Verbindung zu Amazon DocumentDB herstellen, wenn TLS aktiviert ist.

> **Note**
>
> Ab Version 1.2.1 verwendet der MongoDB Go Driver nur das erste CA-Serverzertifikat, das in `sslcertificateauthorityfile` gefunden wurde. Der folgende Beispielcode behebt diese Einschränkung, indem alle in `sslcertificateauthorityfile` gefundenen Serverzertifikate manuell an eine benutzerdefinierte TLS-Konfiguration angehängt werden, die während der Client-Erstellung verwendet wird.

Ersetzen Sie in den folgenden Beispielen jedes *`user input placeholder`* durch die Informationen Ihres Clusters.

```
package main

import (
 "context"
 "fmt"
 "log"
 "time"

 "go.mongodb.org/mongo-driver/bson"
 "go.mongodb.org/mongo-driver/mongo"
 "go.mongodb.org/mongo-driver/mongo/options"

 "io/ioutil"
 "crypto/tls"
 "crypto/x509"
 "errors"
)

const (
 // Path to the AWS CA file
 caFilePath = "global-bundle.pem"

 // Timeout operations after N seconds
 connectTimeout  = 5
 queryTimeout    = 30
 username        = "sample-user"
 password        = "password"
 clusterEndpoint = "sample-cluster.node.us-east-1.docdb.amazonaws.com:27017"

 // Which instances to read from
```

```
 readPreference = "secondaryPreferred"

 connectionStringTemplate = "mongodb://%s:%s@%s/sample-database?
tls=true&replicaSet=rs0&readpreference=%s"
)

func main() {

 connectionURI := fmt.Sprintf(connectionStringTemplate, username, password,
 clusterEndpoint, readPreference)

 tlsConfig, err := getCustomTLSConfig(caFilePath)
 if err != nil {
  log.Fatalf("Failed getting TLS configuration: %v", err)
 }

 client, err :=
 mongo.NewClient(options.Client().ApplyURI(connectionURI).SetTLSConfig(tlsConfig))
 if err != nil {
  log.Fatalf("Failed to create client: %v", err)
 }

 ctx, cancel := context.WithTimeout(context.Background(),
 connectTimeout*time.Second)
 defer cancel()

 err = client.Connect(ctx)
 if err != nil {
  log.Fatalf("Failed to connect to cluster: %v", err)
 }

 // Force a connection to verify our connection string
 err = client.Ping(ctx, nil)
 if err != nil {
  log.Fatalf("Failed to ping cluster: %v", err)
 }

 fmt.Println("Connected to DocumentDB!")

 collection := client.Database("sample-database").Collection("sample-collection")

 ctx, cancel = context.WithTimeout(context.Background(), queryTimeout*time.Second)
 defer cancel()
```

```
 res, err := collection.InsertOne(ctx, bson.M{"name": "pi", "value": 3.14159})
 if err != nil {
  log.Fatalf("Failed to insert document: %v", err)
 }

 id := res.InsertedID
 log.Printf("Inserted document ID: %s", id)

 ctx, cancel = context.WithTimeout(context.Background(), queryTimeout*time.Second)
 defer cancel()

 cur, err := collection.Find(ctx, bson.D{})

 if err != nil {
  log.Fatalf("Failed to run find query: %v", err)
 }
 defer cur.Close(ctx)

 for cur.Next(ctx) {
  var result bson.M
  err := cur.Decode(&result)
  log.Printf("Returned: %v", result)

  if err != nil {
   log.Fatal(err)
  }
 }

 if err := cur.Err(); err != nil {
  log.Fatal(err)
 }

}

func getCustomTLSConfig(caFile string) (*tls.Config, error) {
 tlsConfig := new(tls.Config)
 certs, err := ioutil.ReadFile(caFile)

 if err != nil {
  return tlsConfig, err
 }

 tlsConfig.RootCAs = x509.NewCertPool()
 ok := tlsConfig.RootCAs.AppendCertsFromPEM(certs)
```

```
if !ok {
 return tlsConfig, errors.New("Failed parsing pem file")
}

return tlsConfig, nil
```

Java

Wenn Sie von einer Java-Anwendung aus eine Verbindung zu einem TLS-enabled Amazon DocumentDB-Cluster herstellen, muss Ihr Programm die AWS bereitgestellte Zertifizierungsstelle (CA) verwenden, um die Verbindung zu validieren. Gehen Sie wie folgt vor, um das Amazon RDS-CA-Zertifikat zu verwenden:

1. Laden Sie die Amazon RDS CA-Datei von herunter https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem.
2. Erstellen Sie einen Vertrauensspeicher mit dem in der Datei enthaltenen CA-Zertifikat, indem Sie die folgenden Befehle ausführen. Stellen Sie sicher, dass Sie das *truststore-password* in etwas anderes ändern. Wenn Sie auf einen Vertrauensspeicher zugreifen, der sowohl das alte CA-Zertifikat (`rds-ca-2015-root.pem`) als auch das neue CA-Zertifikat (`rds-ca-2019-root.pem`) enthält, können Sie das Zertifikat-Bundle in den Vertrauensspeicher importieren.

   Nachfolgend finden Sie ein Beispiel-Shell-Skript, das das Zertifikatpaket in einen Trust Store auf einem Linux-Betriebssystem importiert. Ersetzen Sie in den folgenden Beispielen jedes *user input placeholder* durch Ihre eigenen Informationen. Vor allem sollten Sie das Beispielverzeichnis "*mydir*" überall im Skript durch ein Verzeichnis ersetzen, das Sie für diese Aufgabe erstellt haben.

   ```
   mydir=/tmp/certs
   truststore=${mydir}/rds-truststore.jks
   storepassword=truststore-password

   curl -sS "https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem" >
    ${mydir}/global-bundle.pem
   awk 'split_after == 1 {n++;split_after=0} /-----END CERTIFICATE-----/
    {split_after=1}{print > "rds-ca-" n ".pem"}' < ${mydir}/global-bundle.pem

   for CERT in rds-ca-*; do
   ```

```
  alias=$(openssl x509 -noout -text -in $CERT | perl -ne 'next unless /
Subject:/; s/.*(CN=|CN = )//; print')
  echo "Importing $alias"
  keytool -import -file ${CERT} -alias "${alias}" -storepass ${storepassword} -
keystore ${truststore} -noprompt
  rm $CERT
done

rm ${mydir}/global-bundle.pem

echo "Trust store content is: "

keytool -list -v -keystore "$truststore" -storepass ${storepassword} | grep
 Alias | cut -d " " -f3- | while read alias
do
   expiry=`keytool -list -v -keystore "$truststore" -storepass ${storepassword}
 -alias "${alias}" | grep Valid | perl -ne 'if(/until: (.*?)\n/) { print
 "$1\n"; }'`
   echo " Certificate ${alias} expires in '$expiry'"
done
```

Es folgt ein Beispiel für ein Shell-Skript, das das Zertifikatspaket in einen Vertrauensspeicher unter macOS importiert.

```
mydir=/tmp/certs
truststore=${mydir}/rds-truststore.jks
storepassword=truststore-password

curl -sS "https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem" >
 ${mydir}/global-bundle.pem
split -p "-----BEGIN CERTIFICATE-----" ${mydir}/global-bundle.pem rds-ca-

for CERT in rds-ca-*; do
  alias=$(openssl x509 -noout -text -in $CERT | perl -ne 'next unless /
Subject:/; s/.*(CN=|CN = )//; print')
  echo "Importing $alias"
  keytool -import -file ${CERT} -alias "${alias}" -storepass ${storepassword} -
keystore ${truststore} -noprompt
  rm $CERT
done

rm ${mydir}/global-bundle.pem
```

```
echo "Trust store content is: "

keytool -list -v -keystore "$truststore" -storepass ${storepassword} | grep
 Alias | cut -d " " -f3- | while read alias
do
   expiry=`keytool -list -v -keystore "$truststore" -storepass ${storepassword}
 -alias "${alias}" | grep Valid | perl -ne 'if(/until: (.*?)\n/) { print
 "$1\n"; }'`
   echo " Certificate ${alias} expires in '$expiry'"
done
```

3. Verwenden Sie das `keystore` in Ihrem Programm, indem Sie die folgenden Systemeigenschaften in Ihrer Anwendung festlegen, bevor Sie eine Verbindung zum Amazon DocumentDB-Cluster herstellen.

```
javax.net.ssl.trustStore: truststore
javax.net.ssl.trustStorePassword: truststore-password;
```

4. Der folgende Code zeigt, wie Sie mithilfe von Java eine Verbindung zu Amazon DocumentDB herstellen, wenn TLS aktiviert ist.

   Ersetzen Sie im folgenden Beispiel jedes *user input placeholder* durch die Informationen Ihres Clusters.

```
package com.example.documentdb;

import com.mongodb.client.*;
import org.bson.Document;

public final class Test {
    private Test() {
    }
    public static void main(String[] args) {

        String template = "mongodb://%s:%s@%s/sample-database?
ssl=true&replicaSet=rs0&readpreference=%s";
        String username = "sample-user";
        String password = "password";
        String clusterEndpoint = "sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017";
        String readPreference = "secondaryPreferred";
```

```
        String connectionString = String.format(template, username, password,
 clusterEndpoint, readPreference);

        String truststore = "truststore";
        String truststorePassword = "truststore-password";

        System.setProperty("javax.net.ssl.trustStore", truststore);
        System.setProperty("javax.net.ssl.trustStorePassword",
 truststorePassword);

        MongoClient mongoClient = MongoClients.create(connectionString);

        MongoDatabase testDB = mongoClient.getDatabase("sample-database");
        MongoCollection<Document> numbersCollection =
 testDB.getCollection("sample-collection");

        Document doc = new Document("name", "pi").append("value", 3.14159);
        numbersCollection.insertOne(doc);

        MongoCursor<Document> cursor = numbersCollection.find().iterator();
        try {
            while (cursor.hasNext()) {
                System.out.println(cursor.next().toJson());
            }
        } finally {
            cursor.close();
        }

    }
}
```

C# / .NET

Der folgende Code zeigt, wie Sie mithilfe von C#/.NET eine Verbindung zu Amazon DocumentDB herstellen, wenn TLS aktiviert ist.

Ersetzen Sie im folgenden Beispiel jedes *user input placeholder* durch die Informationen Ihres Clusters.

```
using System;
using System.Text;
using System.Linq;
```

```
using System.Collections.Generic;
using System.Security.Cryptography;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using MongoDB.Driver;
using MongoDB.Bson;

namespace DocDB
{
    class Program
    {
        static void Main(string[] args)
        {
            string template = "mongodb://{0}:{1}@{2}/sampledatabase?
tls=true&replicaSet=rs0&readpreference={3}";
            string username = "sample-user";
            string password = "password";
            string readPreference = "secondaryPreferred";
            string clusterEndpoint="sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017";
            string connectionString = String.Format(template, username, password,
 clusterEndpoint, readPreference);

            string pathToCAFile = "<PATH/global-bundle.p7b_file>";

            // ADD CA certificate to local trust store
            // DO this once - Maybe when your service starts
            X509Store localTrustStore = new X509Store(StoreName.Root);
            X509Certificate2Collection certificateCollection = new
 X509Certificate2Collection();
            certificateCollection.Import(pathToCAFile);
            try
            {
                localTrustStore.Open(OpenFlags.ReadWrite);
                localTrustStore.AddRange(certificateCollection);
            }
            catch (Exception ex)
            {
                Console.WriteLine("Root certificate import failed: " + ex.Message);
                throw;
            }
            finally
            {
                localTrustStore.Close();
```

```
            }

            var settings = MongoClientSettings.FromUrl(new
 MongoUrl(connectionString));
            var client = new MongoClient(settings);

            var database = client.GetDatabase("sampledatabase");
            var collection =
 database.GetCollection<BsonDocument>("samplecollection");
            var docToInsert = new BsonDocument { { "pi", 3.14159 } };
            collection.InsertOne(docToInsert);
        }
    }
}
```

MongoDB Shell

Der folgende Code zeigt, wie Sie mit der neuesten Version, Mongosh, oder der vorherigen Mongo-Shell-Version, eine Verbindung zu Amazon DocumentDB herstellen und diese abfragen, wenn TLS aktiviert ist.

Stellen Sie mit Mongosh eine Verbindung zu Amazon DocumentDB her

> **Important**
>
> Es gibt eine bekannte Einschränkung bei Node.js Treibern, die älter als Version 6.13.1 sind und die derzeit von der IAM-Identitätsauthentifizierung für Amazon DocumentDB nicht unterstützt werden. Node.js Treiber und Tools, die Node.js Treiber verwenden (z. B. Mongosh), müssen aktualisiert werden, um die Node.js Treiberversion 6.13.1 oder höher zu verwenden.

Ersetzen Sie in den folgenden Beispielen jedes *`user input placeholder`* durch die Informationen Ihres Clusters.

```
mongosh --tls --host cluster-end-point:27017 --tlsCAFile  global-bundle.pem --
username sample-user --password password --retryWrites false
```

Stellen Sie mit der vorherigen Mongo-Shell-Version eine Verbindung zu Amazon DocumentDB her

Wenn Sie IAM verwenden, müssen Sie eine frühere Version der Mongo-Shell verwenden. Geben Sie eine der folgenden Befehlsoptionen ein:

```
mongo --ssl --host cluster-end-point:27017 --sslCAFile global-bundle.pem --
username sample-user --password password
```

Wenn Sie eine Version verwenden, die mindestens 4.2 entspricht, verwenden Sie den folgenden Code, um eine Verbindung herzustellen. Ab der Engine-Version 8.0.2 unterstützt Amazon DocumentDB wiederholbare Schreibversuche. In früheren Engine-Versionen werden wiederholbare Schreibvorgänge nicht unterstützt. Wenn Sie eine ältere Mongo-Shell (nicht Mongosh) verwenden, fügen Sie den `retryWrites=false` Befehl in keine Codezeichenfolge ein. Weitere Informationen finden Sie unter Wiederholbare Schreibvorgänge in Amazon DocumentDB.

```
mongo --tls --host cluster-end-point:27017 --tlsCAFile global-bundle.pem --
username sample-user --password password
```

Testen Sie die Verbindung

1. Fügen Sie ein einzelnes Dokument ein.

   ```
   db.myTestCollection.insertOne({'hello':'Amazon DocumentDB'})
   ```

2. Suchen Sie das Dokument, das zuvor eingefügt wurde.

   ```
   db.myTestCollection.find({'hello':'Amazon DocumentDB'})
   ```

R

Der folgende Code zeigt, wie Sie mit R mithilfe von mongolite (https://jeroen.github.io/mongolite/) eine Verbindung zu Amazon DocumentDB herstellen, wenn TLS aktiviert ist.

Ersetzen Sie im folgenden Beispiel jedes durch die Informationen *user input placeholder* Ihres Clusters.

```
#Include the mongolite library.
library(mongolite)
```

```
mongourl <- paste("mongodb://sample-user:password@sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017/test2?ssl=true&",
          "readPreference=secondaryPreferred&replicaSet=rs0", sep="")

#Create a MongoDB client, open a connection to Amazon DocumentDB as a replica
#   set and specify the read preference as secondary preferred
client <-  mongo(url = mongourl, options = ssl_options(weak_cert_validation = F, ca
 ="<PATH/global-bundle.pem>"))

#Insert a single document
str <- c('{"hello" : "Amazon DocumentDB"}')
client$insert(str)

#Find the document that was previously written
client$find()
```

Ruby

Der folgende Code zeigt, wie Sie mit Ruby eine Verbindung zu Amazon DocumentDB herstellen, wenn TLS aktiviert ist.

Ersetzen Sie im folgenden Beispiel jedes *user input placeholder* durch die Informationen Ihres Clusters.

```
require 'mongo'
require 'neatjson'
require 'json'
client_host = 'mongodb://sample-cluster.node.us-east-1.docdb.amazonaws.com:27017'
client_options = {
   database: 'test',
   replica_set: 'rs0',
   read: {:secondary_preferred => 1},
   user: 'sample-user',
   password: 'password',
   ssl: true,
   ssl_verify: true,
   ssl_ca_cert: 'PATH/global-bundle.pem',
   retry_writes: false
}


begin
   ##Create a MongoDB client, open a connection to Amazon DocumentDB as a
```

```
   ##   replica set and specify the read preference as secondary preferred
   client = Mongo::Client.new(client_host, client_options)

   ##Insert a single document
   x = client[:test].insert_one({"hello":"Amazon DocumentDB"})

   ##Find the document that was previously written
   result = client[:test].find()

   #Print the document
   result.each do |document|
      puts JSON.neat_generate(document)
   end
end

#Close the connection
client.close
```

Verbindung bei deaktiviertem TLS herstellen

Um ein Codebeispiel für die programmgesteuerte Verbindung zu einem TLS-disabled Amazon DocumentDB-Cluster anzuzeigen, wählen Sie den Tab für die Sprache, die Sie verwenden möchten.

Python

Der folgende Code zeigt, wie Sie mithilfe von Python eine Verbindung zu Amazon DocumentDB herstellen, wenn TLS deaktiviert ist.

Ersetzen Sie im folgenden Beispiel jedes *`user input placeholder`* durch die Informationen Ihres Clusters.

```
## Create a MongoDB client, open a connection to Amazon DocumentDB as a replica set
 and specify the read preference as secondary preferred

import pymongo
import sys

client = pymongo.MongoClient('mongodb://sample-user:password@sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017/?
replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

##Specify the database to be used
```

```
db = client.sample_database

##Specify the collection to be used
col = db.sample_collection

##Insert a single document
col.insert_one({'hello':'Amazon DocumentDB'})

##Find the document that was previously written
x = col.find_one({'hello':'Amazon DocumentDB'})

##Print the result to the screen
print(x)

##Close the connection
client.close()
```

Node.js

Der folgende Code zeigt, wie Sie eine Verbindung zu Amazon DocumentDB herstellen Node.js , wenn TLS deaktiviert ist.

> **Important**
>
> Es gibt eine bekannte Einschränkung bei Node.js Treibern, die älter als Version 6.13.1 sind und die derzeit von der IAM-Identitätsauthentifizierung für Amazon DocumentDB nicht unterstützt werden. Node.js Treiber und Tools, die Node.js Treiber verwenden (z. B. Mongosh), müssen aktualisiert werden, um die Node.js Treiberversion 6.13.1 oder höher zu verwenden.

Ersetzen Sie im folgenden Beispiel jedes durch die Informationen *user input placeholder* Ihres Clusters.

```
var MongoClient = require('mongodb').MongoClient;

//Create a MongoDB client, open a connection to Amazon DocumentDB as a replica set,
//  and specify the read preference as secondary preferred
var client = MongoClient.connect(
'mongodb://sample-user:password@sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017/sample-database?
replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false',
```

```
{
  useNewUrlParser: true
},

function(err, client) {
    if(err)
        throw err;
    //Specify the database to be used
    db = client.db('sample-database');

    //Specify the collection to be used
    col = db.collection('sample-collection');

    //Insert a single document
    col.insertOne({'hello':'Amazon DocumentDB'}, function(err, result){
      //Find the document that was previously written
      col.findOne({'hello':'Amazon DocumentDB'}, function(err, result){
        //Print the result to the screen
        console.log(result);

        //Close the connection
        client.close()
      });
   });
});
```

PHP

Der folgende Code zeigt, wie Sie mithilfe von PHP eine Verbindung zu Amazon DocumentDB herstellen, wenn TLS deaktiviert ist.

Ersetzen Sie im folgenden Beispiel jedes *user input placeholder* durch die Informationen Ihres Clusters.

```
<?php
//Include Composer's autoloader
require 'vendor/autoload.php';

//Create a MongoDB client and open connection to Amazon DocumentDB
$client = new MongoDB\Client("mongodb://sample-user:password@sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017/?retryWrites=false");

//Specify the database and collection to be used
```

```
$col = $client->sampledatabase->samplecollection;

//Insert a single document
$result = $col->insertOne( [ 'hello' => 'Amazon DocumentDB'] );

//Find the document that was previously written
$result = $col->findOne(array('hello' => 'Amazon DocumentDB'));

//Print the result to the screen
print_r($result);
?>
```

Go

Der folgende Code zeigt, wie Sie mithilfe von Go eine Verbindung zu Amazon DocumentDB herstellen, wenn TLS deaktiviert ist.

Ersetzen Sie im folgenden Beispiel jedes *user input placeholder* durch die Informationen Ihres Clusters.

```
package main

import (
 "context"
 "fmt"
 "log"
 "time"

 "go.mongodb.org/mongo-driver/bson"
 "go.mongodb.org/mongo-driver/mongo"
 "go.mongodb.org/mongo-driver/mongo/options"
)

const (
 // Timeout operations after N seconds
 connectTimeout  = 5
 queryTimeout    = 30
 username        = "sample-user"
 password        = "password"
 clusterEndpoint = "sample-cluster.node.us-east-1.docdb.amazonaws.com:27017"

 // Which instances to read from
 readPreference           = "secondaryPreferred"
```

```
 connectionStringTemplate = "mongodb://%s:%s@%s/sample-database?
replicaSet=rs0&readpreference=%s"
)

func main() {

 connectionURI := fmt.Sprintf(connectionStringTemplate, username, password,
 clusterEndpoint, readPreference)

 client, err := mongo.NewClient(options.Client().ApplyURI(connectionURI))
 if err != nil {
  log.Fatalf("Failed to create client: %v", err)
 }

 ctx, cancel := context.WithTimeout(context.Background(),
 connectTimeout*time.Second)
 defer cancel()

 err = client.Connect(ctx)
 if err != nil {
  log.Fatalf("Failed to connect to cluster: %v", err)
 }

 // Force a connection to verify our connection string
 err = client.Ping(ctx, nil)
 if err != nil {
  log.Fatalf("Failed to ping cluster: %v", err)
 }

 fmt.Println("Connected to DocumentDB!")

 collection := client.Database("sample-database").Collection("sample-collection")

 ctx, cancel = context.WithTimeout(context.Background(), queryTimeout*time.Second)
 defer cancel()

 res, err := collection.InsertOne(ctx, bson.M{"name": "pi", "value": 3.14159})
 if err != nil {
  log.Fatalf("Failed to insert document: %v", err)
 }

 id := res.InsertedID
 log.Printf("Inserted document ID: %s", id)
```

```
 ctx, cancel = context.WithTimeout(context.Background(), queryTimeout*time.Second)
 defer cancel()

 cur, err := collection.Find(ctx, bson.D{})

 if err != nil {
  log.Fatalf("Failed to run find query: %v", err)
 }
 defer cur.Close(ctx)

 for cur.Next(ctx) {
  var result bson.M
  err := cur.Decode(&result)
  log.Printf("Returned: %v", result)

  if err != nil {
   log.Fatal(err)
  }
 }

 if err := cur.Err(); err != nil {
  log.Fatal(err)
 }

}
```

Java

Der folgende Code zeigt, wie Sie mithilfe von Java eine Verbindung zu Amazon DocumentDB herstellen, wenn TLS deaktiviert ist.

Ersetzen Sie im folgenden Beispiel jedes *`user input placeholder`* durch die Informationen Ihres Clusters.

```
package com.example.documentdb;

import com.mongodb.MongoClient;
import com.mongodb.MongoClientURI;
import com.mongodb.ServerAddress;
import com.mongodb.MongoException;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.MongoCollection;
```

```
import org.bson.Document;


public final class Main {
    private Main() {
    }
    public static void main(String[] args) {

        String template = "mongodb://%s:%s@%s/sample-database?
replicaSet=rs0&readpreference=%s";
        String username = "sample-user";
        String password = "password";
        String clusterEndpoint = "sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017";
        String readPreference = "secondaryPreferred";
        String connectionString = String.format(template, username, password,
 clusterEndpoint, readPreference);

        MongoClientURI clientURI = new MongoClientURI(connectionString);
        MongoClient mongoClient = new MongoClient(clientURI);

        MongoDatabase testDB = mongoClient.getDatabase("sample-database");
        MongoCollection<Document> numbersCollection = testDB.getCollection("sample-
collection");

        Document doc = new Document("name", "pi").append("value", 3.14159);
        numbersCollection.insertOne(doc);

        MongoCursor<Document> cursor = numbersCollection.find().iterator();
        try {
            while (cursor.hasNext()) {
                System.out.println(cursor.next().toJson());
            }
        } finally {
            cursor.close();
        }

    }
}
```

C# / .NET

Der folgende Code zeigt, wie Sie mithilfe von C#/.NET eine Verbindung zu Amazon DocumentDB herstellen, wenn TLS deaktiviert ist.

Ersetzen Sie im folgenden Beispiel jedes *user input placeholder* durch die Informationen Ihres Clusters.

```
using System;
using System.Text;
using System.Linq;
using System.Collections.Generic;
using System.Security.Cryptography;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using MongoDB.Driver;
using MongoDB.Bson;

namespace CSharpSample
{
   class Program
    {
       static void Main(string[] args)
        {
           string template = "mongodb://{0}:{1}@{2}/sampledatabase?
replicaSet=rs0&readpreference={3}";
           string username = "sample-user";
           string password = "password";
           string clusterEndpoint = "sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017";
           string readPreference = "secondaryPreferred";
           string connectionString = String.Format(template, username, password,
 clusterEndpoint, readPreference);

           var settings = MongoClientSettings.FromUrl(new
 MongoUrl(connectionString));
           var client = new MongoClient(settings);

           var database = client.GetDatabase("sampledatabase");
           var collection =
 database.GetCollection<BsonDocument>("samplecollection");
           var docToInsert = new BsonDocument { { "pi", 3.14159 } };
            collection.InsertOne(docToInsert);
        }
    }
}
```

MongoDB Shell

Der folgende Code zeigt, wie Sie mit der neuesten Version, Mongosh, oder der vorherigen Mongo-Shell-Version eine Verbindung zu Amazon DocumentDB herstellen und diese abfragen, wenn TLS deaktiviert ist.

Stellen Sie mit Mongosh eine Verbindung zu Amazon DocumentDB her

> **Important**
>
> Es gibt eine bekannte Einschränkung bei Node.js Treibern, die älter als Version 6.13.1 sind und die derzeit von der IAM-Identitätsauthentifizierung für Amazon DocumentDB nicht unterstützt werden. Node.js Treiber und Tools, die Node.js Treiber verwenden (z. B. Mongosh), müssen aktualisiert werden, um die Node.js Treiberversion 6.13.1 oder höher zu verwenden.

Ersetzen Sie in den folgenden Beispielen jedes *user input placeholder* durch die Informationen Ihres Clusters.

```
mongosh --host cluster-end-point:27017 --username sample-user --password password --
retryWrites false
```

Stellen Sie mit der vorherigen Mongo-Shell-Version eine Verbindung zu Amazon DocumentDB her

Wenn Sie IAM verwenden, müssen Sie eine frühere Version der Mongo-Shell verwenden. Geben Sie eine der folgenden Befehlsoptionen ein:

```
mongo --host cluster-end-point:27017 --username sample-user --password password
```

Wenn Sie eine Version verwenden, die mindestens 4.2 entspricht, verwenden Sie den folgenden Code, um eine Verbindung herzustellen. Wiederholbare Schreibvorgänge werden in Amazon DocumentDB nicht unterstützt. Wenn Sie eine ältere Mongo-Shell (nicht Mongosh) verwenden, fügen Sie den `retryWrites=false` Befehl in keine Codezeichenfolge ein. Standardmäßig sind wiederholbare Schreibvorgänge deaktiviert. `retryWrites=false`Das Einschließen kann zu einem Fehler bei normalen Lesebefehlen führen.

```
mongo --host cluster-end-point:27017 --username sample-user --password password
```

Testen Sie die Verbindung

1. Fügen Sie ein einzelnes Dokument ein.

```
db.myTestCollection.insertOne({'hello':'Amazon DocumentDB'})
```

2. Suchen Sie das Dokument, das zuvor eingefügt wurde.

```
db.myTestCollection.find({'hello':'Amazon DocumentDB'})
```

R

Der folgende Code zeigt, wie Sie mit R mithilfe von mongolite (https://jeroen.github.io/mongolite/) eine Verbindung zu Amazon DocumentDB herstellen, wenn TLS deaktiviert ist.

Ersetzen Sie im folgenden Beispiel jedes durch die Informationen *user input placeholder* Ihres Clusters.

```
#Include the mongolite library.
library(mongolite)

#Create a MongoDB client, open a connection to Amazon DocumentDB as a replica
#   set and specify the read preference as secondary preferred
client <- mongo(url = "mongodb://sample-user:password@sample-
cluster.node.us-east-1.docdb.amazonaws.com:27017/sample-database?
readPreference=secondaryPreferred&replicaSet=rs0")

##Insert a single document
str <- c('{"hello" : "Amazon DocumentDB"}')
client$insert(str)

##Find the document that was previously written
client$find()
```

Ruby

Der folgende Code zeigt, wie Sie mit Ruby eine Verbindung zu Amazon DocumentDB herstellen, wenn TLS deaktiviert ist.

Ersetzen Sie im folgenden Beispiel jedes *user input placeholder* durch die Informationen Ihres Clusters.

```
require 'mongo'
require 'neatjson'
require 'json'
client_host = 'mongodb://sample-cluster.node.us-east-1.docdb.amazonaws.com:27017'
client_options = {
   database: 'test',
   replica_set: 'rs0',
   read: {:secondary_preferred => 1},
   user: 'sample-user',
   password: 'password',
   retry_writes: false
}

begin
   ##Create a MongoDB client, open a connection to Amazon DocumentDB as a
   ##   replica set and specify the read preference as secondary preferred
   client = Mongo::Client.new(client_host, client_options)

   ##Insert a single document
   x = client[:test].insert_one({"hello":"Amazon DocumentDB"})

   ##Find the document that was previously written
   result = client[:test].find()

   #Print the document
   result.each do |document|
      puts JSON.neat_generate(document)
   end
end

#Close the connection
client.close
```

Netzwerkkomprimierung in Amazon DocumentDB

Mit der Wire-Protocol-Komprimierung können Sie die Daten reduzieren, die Ihre Anwendung über das Netzwerk sendet. Amazon DocumentDB 8.0.1 und höher unterstützen die Komprimierung. Sie erhalten einen verbesserten Durchsatz für Workloads, die große Dokumente oder große Ergebnismengen übertragen. Nachdem Sie die Komprimierung aktiviert haben, tauschen Treiber und Server komprimierte Nachrichten aus.

Wenn Sie die Netzwerkkomprimierung aktivieren, tauschen Sie CPU-Ressourcen gegen eine reduzierte Netzwerkbandbreite ein. Aktivieren Sie die Komprimierung, wenn Ihr Workload netzwerkgebunden ist und große Nutzlasten überträgt. Bei kleinen Anfragen und Antworten sehen Sie möglicherweise keinen Nettogewinn, da der Komprimierungsaufwand die Einsparungen übersteigen kann.

Unterstützte Kompressoren

Amazon DocumentDB unterstützt den `snappy` Kompressor. Amazon DocumentDB unterstützt keine anderen Kompressoren, die von MongoDB-Treibern angeboten werden, wie z. B. und. `zlib` `zstd`

Wenn ein Treiber eine Liste von Kompressoren anfordert, handelt Amazon DocumentDB nur die unterstützten Kompressoren aus. Wenn beispielsweise ein Treiber `snappy` und anbietet, wählt `snappy` Amazon DocumentDB diese aus und `zstd` ignoriert sie. `zstd` Wenn ein Treiber nur Kompressoren anbietet, die nicht unterstützt werden, komprimiert Amazon DocumentDB die Verbindung nicht. In diesem Fall tauschen der Treiber und der Server Nachrichten ohne Komprimierung aus.

Aktivieren Sie die Komprimierung

Um die Komprimierung zu aktivieren, fügen Sie die `compressors` Option zu Ihrer Verbindungszeichenfolge hinzu und setzen Sie sie auf`snappy`. Ersetzen Sie *`username`* in der folgenden Verbindungszeichenfolge und *`docdb-cluster-endpoint`* durch Ihre eigenen Werte: *`password`*

```
mongodb://username:password@docdb-cluster-endpoint:27017/?
tls=true&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false&compressors=snappy
```

Wenn Sie einen MongoDB-Treiber verwenden, stellen Sie dieselbe Option in den Verbindungsoptionen des Treibers ein. Die meisten Treiber benötigen eine zusätzliche clientseitige Abhängigkeit, um eine schnelle Komprimierung durchzuführen:

- Der Node.js Treiber benötigt das Paket. `snappy`
- Der Python-Treiber (PyMongo) benötigt das `python-snappy` Paket. Unter Linux ist `python-snappy` außerdem die Systembibliothek Snappy erforderlich (z. B. `snappy-devel` through `dnf` oryum).

Den genauen Optionsnamen und alle erforderlichen Pakete finden Sie in der Dokumentation Ihres Treibers.

> **Note**
>
> Wenn die clientseitige Komprimierungsabhängigkeit fehlt, stellt der Treiber erfolgreich eine Verbindung her. Er tauscht jedoch Nachrichten ohne Komprimierung aus und gibt keinen Fehler zurück. Um zu bestätigen, dass die Komprimierung aktiv ist, überprüfen Sie den ausgehandelten Kompressor, wie unter beschriebenÜberprüfen Sie die Komprimierung.

Das folgende Python-Codebeispiel stellt eine Verbindung zu Ihrem Cluster her, wobei die schnelle Komprimierung aktiviert ist, und verwendet einen `try except` AND-Block, um Verbindungsfehler zu behandeln:

```
import pymongo

try:
    client = pymongo.MongoClient(
        'mongodb://username:password@docdb-cluster-endpoint:27017/'
        '?tls=true&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false'
        '&compressors=snappy',
        tlsCAFile='global-bundle.pem')

    db = client['sample-database']
    col = db['sample-collection']
    col.insert_one({'item': 'DocumentDB'})
    print(col.find_one())
except Exception as error:
    print(f'Connection failed: {error}')
```

Überprüfen Sie die Komprimierung

Um zu bestätigen, dass ein Treiber und Amazon DocumentDB die Komprimierung ausgehandelt haben, überprüfen Sie das `compression` Feld, das der `hello` MongoDB-Wire-Protokollbefehl zurückgibt. Wenn der Treiber und Amazon DocumentDB die Komprimierung aushandeln, werden in diesem Feld die vereinbarten Kompressoren aufgeführt.

Verbindung zu einem Amazon DocumentDB-Cluster von außerhalb einer Amazon VPC herstellen

Amazon DocumentDB-Cluster (mit MongoDB-Kompatibilität) werden in einer Amazon Virtual Private Cloud (Amazon VPC) bereitgestellt. Auf sie kann direkt von Amazon EC2-Instances oder anderen

AWS Diensten zugegriffen werden, die in derselben Amazon VPC bereitgestellt werden. Darüber hinaus kann über VPC-Peering von EC2-Instances oder anderen AWS Diensten in verschiedenen VPCs in derselben AWS-Region oder anderen Regionen auf Amazon DocumentDB zugegriffen werden.

Nehmen wir jedoch an, Ihr Anwendungsfall erfordert, dass Sie (oder Ihre Anwendung) von außerhalb der VPC des Clusters auf Ihre Amazon DocumentDB-Ressourcen zugreifen. In diesem Fall können Sie SSH-Tunneling (auch bekannt als Portweiterleitung) verwenden, um auf Ihre Amazon DocumentDB-Ressourcen zuzugreifen.

Die eingehende Diskussion zum Thema SSH-Tunneling geht über den Rahmen dieses Themas hinaus. Weitere Informationen zum SSH-Tunneling finden Sie in den folgenden Ressourcen:

- SSH-Tunnel
- Beispiel zum SSH-Port-Forwarding, insbesondere der Abschnitt Lokales Forwarding.

Um einen SSH-Tunnel zu erstellen, benötigen Sie eine Amazon EC2-Instance, die in derselben Amazon VPC wie Ihr Amazon DocumentDB-Cluster ausgeführt wird. Sie können entweder eine vorhandene EC2-Instance in derselben VPC wie Ihr Cluster verwenden oder eine erstellen. Weitere Informationen finden Sie im entsprechenden Thema für Ihr Betriebssystem:

- Erste Schritte mit Amazon EC2 Linux-Instances
- Erste Schritte mit Amazon EC2-Windows-Instances

Normalerweise verwenden Sie vermutlich den folgenden Befehl, um sich mit einer EC2-Instance zu verbinden:

```
ssh -i "ec2Access.pem" ubuntu@ec2-34-229-221-164.compute-1.amazonaws.com
```

In diesem Fall können Sie einen SSH-Tunnel zum Amazon DocumentDB-Cluster einrichten, `sample-cluster.node.us-east-1.docdb.amazonaws.com` indem Sie den folgenden Befehl auf Ihrem lokalen Computer ausführen. Das `-L`-Flag dient zur Weiterleitung eines lokalen Ports. Wenn Sie einen SSH-Tunnel verwenden, stellen Sie über den Cluster-Endpunkt eine Verbindung zu Ihrem Cluster her und versuchen Sie nicht, eine Verbindung im Replikatsatzmodus herzustellen (d. h., indem Sie dies `replicaSet=rs0` in Ihrer Verbindungszeichenfolge angeben), da dies zu einem Fehler führen würde.

```
ssh -i "ec2Access.pem" -L 27017:sample-cluster.node.us-east-1.docdb.amazonaws.com:27017
 ubuntu@ec2-34-229-221-164.compute-1.amazonaws.com -N
```

Nachdem der SSH-Tunnel erstellt wurde, werden alle Befehle, die Sie dazu `localhost:27017` ausgeben, an den Amazon DocumentDB-Cluster weitergeleitet, der in der Amazon `sample-cluster` VPC ausgeführt wird. Wenn Transport Layer Security (TLS) auf Ihrem Amazon DocumentDB-Cluster aktiviert ist, müssen Sie den öffentlichen Schlüssel für Amazon DocumentDB von herunterladen. https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem Der folgende Vorgang lädt diese Datei herunter:

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

> **Note**
>
> TLS ist standardmäßig für neue Amazon DocumentDB-Cluster aktiviert. Eine Deaktivierung ist jedoch möglich. Weitere Informationen finden Sie unter Verwaltung der TLS-Einstellungen für Amazon DocumentDB-Cluster.

Verwenden Sie den folgenden Befehl, um von außerhalb der Amazon VPC eine Verbindung zu Ihrem Amazon DocumentDB-Cluster herzustellen.

```
mongo --sslAllowInvalidHostnames --ssl --sslCAFile global-bundle.pem --username
 <yourUsername> --password <yourPassword>
```

Verbindung zu Amazon DocumentDB als Replikatsatz herstellen

Wenn Sie für Amazon DocumentDB entwickeln, stellen Sie als Replikatgruppe eine Verbindung zu Ihrem Cluster her und verteilen Sie Lesevorgänge mithilfe der integrierten Lesepräferenzfunktionen Ihres Treibers an Replica-Instances. In diesem Abschnitt wird näher erläutert, was das bedeutet, und es wird beschrieben, wie Sie am Beispiel von Python eine Verbindung zu Ihrem Amazon DocumentDB-Cluster als Replikatsatz herstellen können.

Amazon DocumentDB hat drei Endpunkte, die Sie verwenden können, um eine Verbindung zu Ihrem Cluster herzustellen:

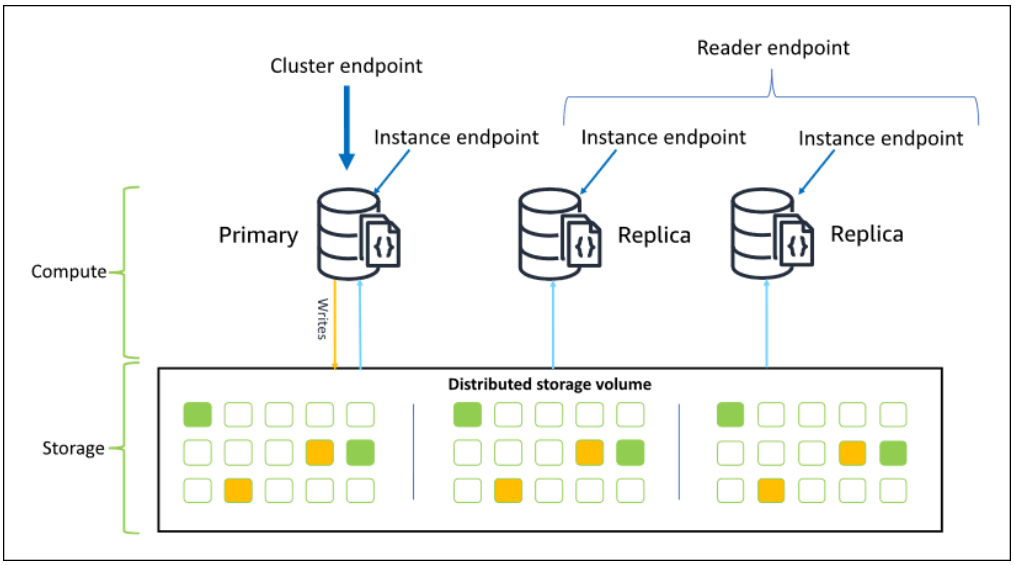
- Cluster-Endpunkt

- Leser-Endpunkt
- Instance-Endpunkte

Verwenden Sie den Cluster-Endpunkt, wenn Sie eine Verbindung zu Amazon DocumentDB herstellen. Dies ist ein CNAME, der auf die primäre Instance in Ihrem Cluster verweist wie im folgenden Diagramm gezeigt.

Wenn Sie einen SSH-Tunnel verwenden, stellen Sie über den Cluster-Endpunkt eine Verbindung zu Ihrem Cluster her. Stellen Sie keine Verbindung im Replikatsatzmodus her (geben Sie dies `replicaSet=rs0` in Ihrer Verbindungszeichenfolge an), da dies zu einem Fehler führen würde.

> **Note**
>
> Weitere Informationen zu Amazon DocumentDB-Endpunkten finden Sie unter. Amazon DocumentDB-Endpunkte



Mithilfe des Cluster-Endpunkts können Sie im Replikatsatzmodus eine Verbindung zu Ihrem Cluster herstellen. Anschließend können Sie die integrierten Treiberfunktionen für Leseeinstellungen verwenden. Im folgenden Beispiel signalisiert die Angabe von `/?replicaSet=rs0` dem SDK, das Sie eine Verbindung als Replikatsatz herstellen möchten. Wenn Sie `/?replicaSet=rs0'` auslassen, leitet der Client alle Anfragen an den Cluster-Endpunkt weiter, d. h. Ihre primäre Instance.

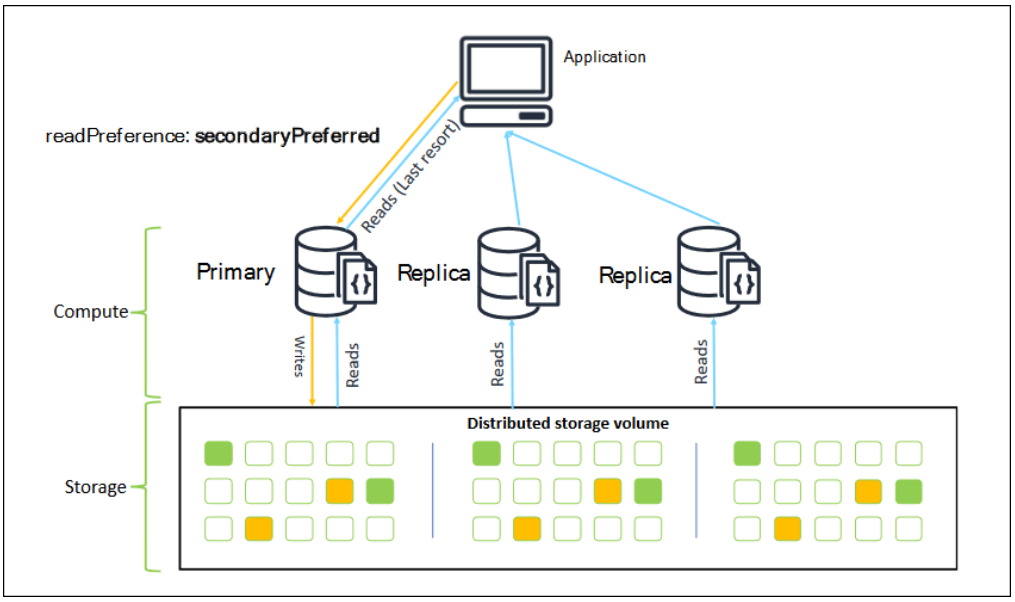
```
## Create a MongoDB client, open a connection to Amazon DocumentDB as a
   ## replica set and specify the read preference as secondary preferred
```

```
   client = pymongo.MongoClient('mongodb://<user-name>:<password>@mycluster.node.us-
east-1.docdb.amazonaws.com:27017/?replicaSet=rs0')
```

Der Vorteil der Verbindung als Replikatsatz besteht darin, dass Ihr SDK die Cluster-Topographie automatisch erkennt, auch wenn Instances dem Cluster hinzugefügt oder aus dem Cluster entfernt werden. Anschließend können Sie den Cluster effizienter nutzen, indem Sie Leseanforderungen an Ihre Replikat-Instances weiterleiten.

Wenn Sie eine Verbindung als Replikatsatz herstellen, können Sie die `readPreference` für die Verbindung angeben. Wenn Sie die Leseeinstellung `secondaryPreferred` angeben, leitet der Client Leseabfragen an Ihre Replikate und Schreibabfragen an Ihre primäre Instance weiter (wie im folgenden Diagramm gezeigt). Damit werden Ihre Cluster-Ressourcen besser genutzt. Weitere Informationen finden Sie unter Präferenzoptionen lesen.

```
## Create a MongoDB client, open a connection to Amazon DocumentDB as a
   ##   replica set and specify the read preference as secondary preferred
client = pymongo.MongoClient('mongodb://<user-name>:<password>@mycluster.node.us-
east-1.docdb.amazonaws.com:27017/?replicaSet=rs0&readPreference=secondaryPreferred')
```



Lesevorgänge aus Amazon DocumentDB-Replikaten sind letztendlich konsistent. Sie geben die Daten in der gleichen Reihenfolge zurück, in der sie auf dem primären Knoten geschrieben wurden. Die Replikationsverzögerung beträgt häufig weniger als 50 ms. Sie können die Replikatverzögerung für Ihren Cluster mithilfe der CloudWatch Amazon-Metriken und überwachen. `DBInstanceReplicaLag` `DBClusterReplicaLagMaximum` Weitere Informationen finden Sie unter Überwachung von Amazon DocumentDB mit CloudWatch.

Im Gegensatz zur herkömmlichen monolithischen Datenbankarchitektur trennt Amazon DocumentDB Speicher und Rechenleistung. Aufgrund dieser modernen Architektur sollten Sie Lesevorgänge auf Replikat-Instances skalieren. Lesevorgänge auf Replikat-Instances blockieren keine Schreibvorgänge, die von der primären Instance repliziert werden. Sie können einem Cluster bis zu 15 Read-Reaplica-Instances hinzufügen und auf Millionen von Lesevorgängen pro Sekunde skalieren.

Der Hauptvorteil der Verbindung als Replikatsatz und der Verteilung von Lesevorgängen an Replikate besteht darin, dass dies die Gesamtzahl der Ressourcen im Cluster erhöht, die für Ihre Anwendung verfügbar sind. Es hat sich bewährt, eine Verbindung als Replikatsatz herzustellen. Dies ist in den folgenden Szenarien am vorteilhaftesten:

- Sie nutzen nahezu 100 Prozent der CPU-Leistung auf Ihrer primären Instance.
- Das Puffer-Cache-Treffer-Verhältnis ist beinahe Null.
- Sie erreichen die Verbindungs- oder Cursor-Limits für eine einzelne Instance.

Die Aufwärtsskalierung einer Cluster-Instance-Größe ist eine Option und kann in einigen Fällen die beste Möglichkeit für die Skalierung des Clusters darstellen. Sie sollten jedoch auch überlegen, wie Sie die Replikate, die in Ihrem Cluster bereits vorhanden sind, besser nutzen können. So können Sie die Skalierung erhöhen, ohne dass für Sie höhere Kosten für die Verwendung eines größeren Instance-Typs anfallen. Wir empfehlen außerdem, diese Grenzwerte (d. h.`CPUUtilization`, `DatabaseConnections` und`BufferCacheHitRatio`) mithilfe von CloudWatch Alarmen zu überwachen und Warnmeldungen zu verwenden, damit Sie wissen, wann eine Ressource stark beansprucht wird.

Weitere Informationen finden Sie unter den folgenden Themen:

- Bewährte Methoden für Amazon DocumentDB
- Amazon DocumentDB-Kontingente

Verwenden von Cluster-Verbindungen

Betrachten Sie ein Szenario, in dem alle Verbindungen in Ihrem Cluster genutzt werden. Beispielsweise hat eine `r5.2xlarge`-Instance ein Limit von 4.500 Verbindungen (und 450 offenen Cursors). Wenn Sie einen Amazon DocumentDB-Cluster mit drei Instanzen erstellen und über den Cluster-Endpunkt nur eine Verbindung zur primären Instance herstellen, liegen Ihre Cluster-Grenzwerte für offene Verbindungen und Cursor bei 4.500 bzw. 450. Sie können diese Limits

erreichen, wenn Sie Anwendungen mit zahlreichen Workern verwenden, die in Containern gestartet werden. Die Container öffnen mehrere Verbindungen gleichzeitig und sättigen den Cluster.

Stattdessen könnten Sie als Replikatgruppe eine Verbindung zum Amazon DocumentDB-Cluster herstellen und Ihre Lesevorgänge an die Replikat-Instances verteilen. Anschließend könnten Sie die Anzahl der verfügbaren Verbindungen und Cursors im Cluster effektiv auf 13.500 bzw. 1.350 verdreifachen. Das Hinzufügen weiterer Instances zum Cluster erhöht lediglich die Anzahl der Verbindungen und Cursors für Lese-Workloads. Wenn Sie die Anzahl der Verbindungen für Schreibvorgänge in Ihrem Cluster erhöhen müssen, sollten Sie die Instance-Größe erhöhen.

> **Note**
>
> Die Anzahl der Verbindungen für `large`, `xlarge`, und `2xlarge` Instances erhöht sich mit der Instance-Größe bis zu 4.500. Die maximale Anzahl von Verbindungen pro Instance für `4xlarge` Instances oder größer beträgt 4.500. Weitere Hinweise zu Beschränkungen nach Instance-Typ finden Sie unter Kontingente für Instanzen.

In der Regel empfehlen wir das Herstellen von Verbindungen mit Ihrem Cluster unter Verwendung der Leseeinstellung `secondary` nicht. Der Grund hierfür ist, dass die Lesevorgänge fehlschlagen, wenn es keine Replikat-Instances im Cluster gibt. Nehmen wir zum Beispiel an, Sie haben einen Amazon DocumentDB-Cluster mit zwei Instanzen, einer Primär- und einer Replica. Wenn die Replikat-Instance ein Problem aufweist, schlagen Leseanforderungen aus einem als `secondary` festgelegten Verbindungspool fehl. Der Vorteil von `secondaryPreferred` besteht darin, dass der Client für Lesevorgänge auf den primären Knoten zurückgreift, wenn er keine geeignete Replikat-Instance für die Verbindung finden kann.

Mehrere Verbindungspools

In einigen Szenarien müssen Lesevorgänge in einer Anwendung eine Lese-nach-Schreib-Konsistenz aufweisen, die nur von der primären Instance in Amazon DocumentDB aus bedient werden kann. In diesen Szenarien könnten Sie zwei Client-Verbindungspools erstellen: einen für Schreibvorgänge und einen für Lesevorgänge, die „Read after Write Consistency“ benötigen. Ihr Code würde in diesem Fall ungefähr wie folgt aussehen.

```
## Create a MongoDB client,
##    open a connection to Amazon DocumentDB as a replica set and specify the
 readPreference as primary
```

```
clientPrimary = pymongo.MongoClient('mongodb://<user-
name>:<password>@mycluster.node.us-east-1.docdb.amazonaws.com:27017/?
replicaSet=rs0&readPreference=primary')

## Create a MongoDB client,
##    open a connection to Amazon DocumentDB as a replica set and specify the
 readPreference as secondaryPreferred
secondaryPreferred = pymongo.MongoClient('mongodb://<user-
name>:<password>@mycluster.node.us-east-1.docdb.amazonaws.com:27017/?
replicaSet=rs0&readPreference=secondaryPreferred')
```

Eine weitere Möglichkeit besteht darin, einen einzelnen Verbindungspool zu erstellen und die Leseeinstellung für eine bestimmte Sammlung zu überschreiben.

```
##Specify the collection and set the read preference level for that collection
col = db.review.with_options(read_preference=ReadPreference.SECONDARY_PREFERRED)
```

Zusammenfassung

Um die Ressourcen in Ihrem Cluster besser nutzen zu können, stellen Sie die Verbindung im Replikat-Set-Modus her. Sie können die Zahl der Lesevorgänge für Ihre Anwendung skalieren, indem Sie Ihre Lesevorgänge an die Replikat-Instances verteilen, wenn dies für Ihre Anwendung geeignet ist.

Von Studio 3T aus eine Verbindung zu einem Amazon DocumentDB-Cluster herstellen

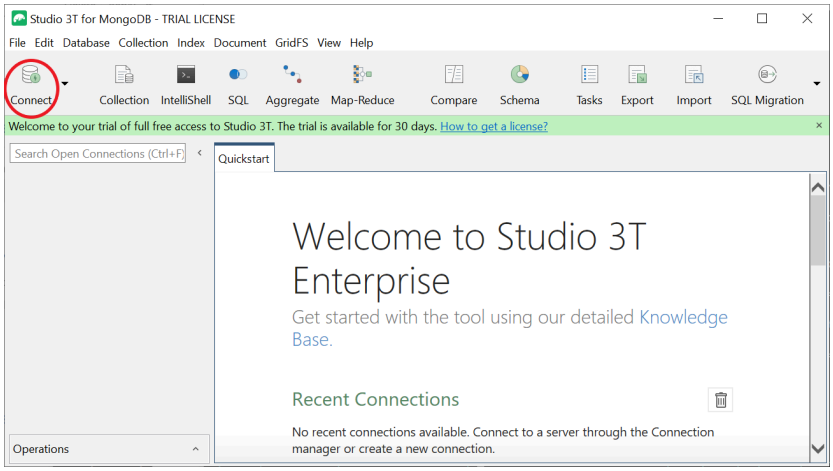
Studio 3T ist eine beliebte GUI und IDE für Entwickler und Dateningenieure, die mit MongoDB arbeiten. Es bietet mehrere leistungsstarke Funktionen Baum-, Tabellen- und JSON-Ansichten Ihrer Daten, einfach import/export in CSV, JSON, SQL und flexible Abfrageoptionen BSON/mongodump, eine visuelle Drag-and-Drop-Benutzeroberfläche, eine integrierte Mongo-Shell mit automatischer Vervollständigung, einen Aggregationspipeline-Editor und SQL-Abfrageunterstützung.

Voraussetzungen

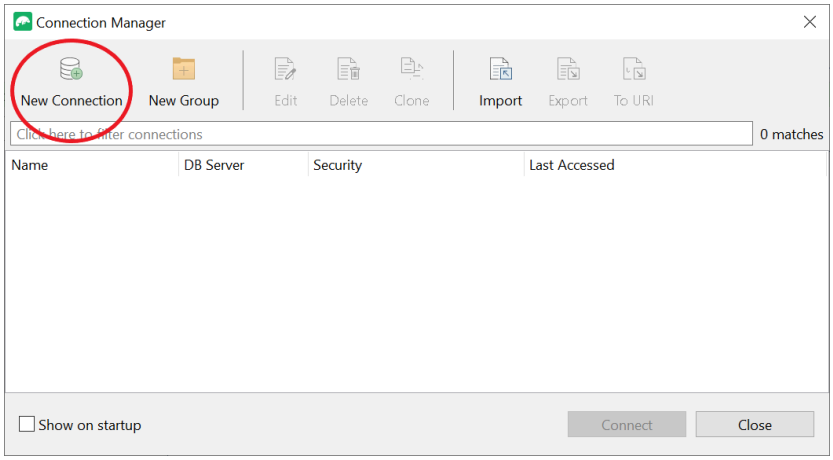
- Wenn Sie noch keinen Amazon DocumentDB-Cluster haben, der Amazon EC2 als bastion/jump Host verwendet, finden Sie weitere Informationen unter. Stellen Sie mithilfe von Amazon EC2 eine Verbindung her
- Wenn Sie Studio 3T nicht haben, laden Sie es herunter und installieren Sie es.

Mit Studio 3T Connect

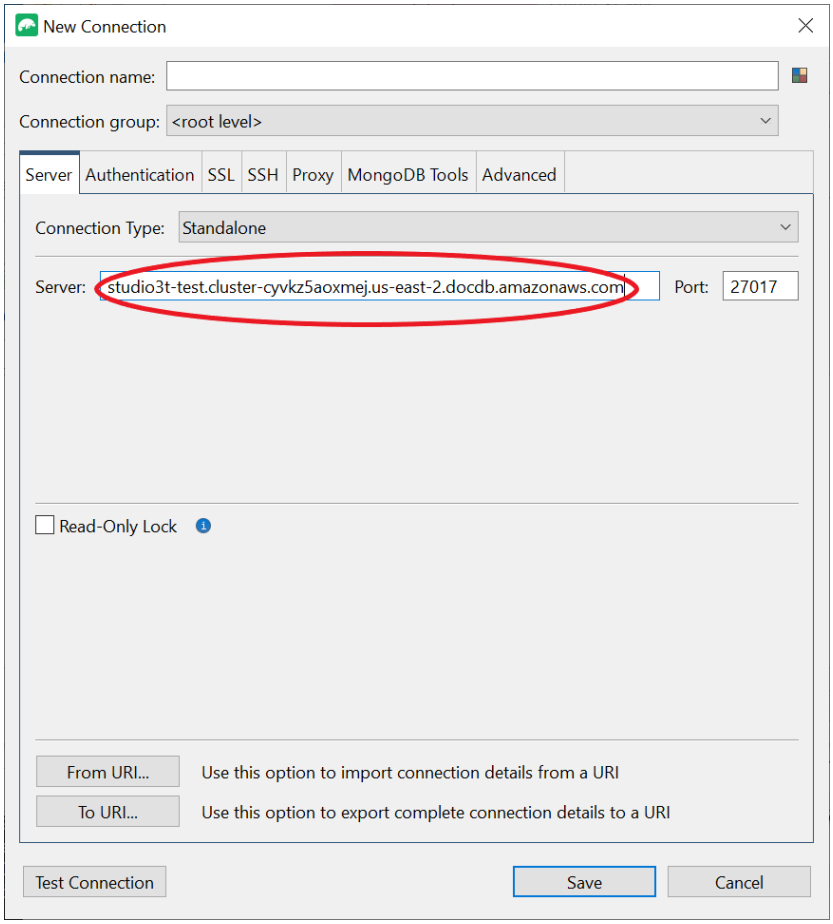
1. Wählen Sie in der oberen linken Ecke der Werkzeugleiste Connect.



2. Wählen Sie in der oberen linken Ecke der Werkzeugleiste Neue Verbindung.

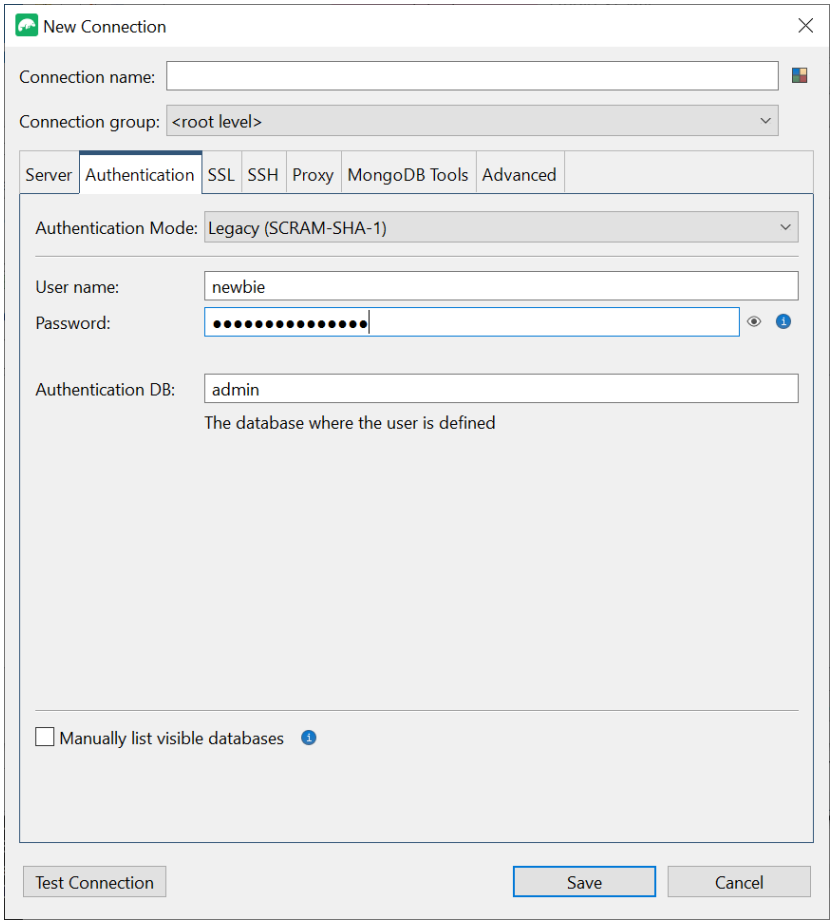


3. Geben Sie auf der Registerkarte Server im Feld Server die Informationen zum Cluster-Endpunkt ein.

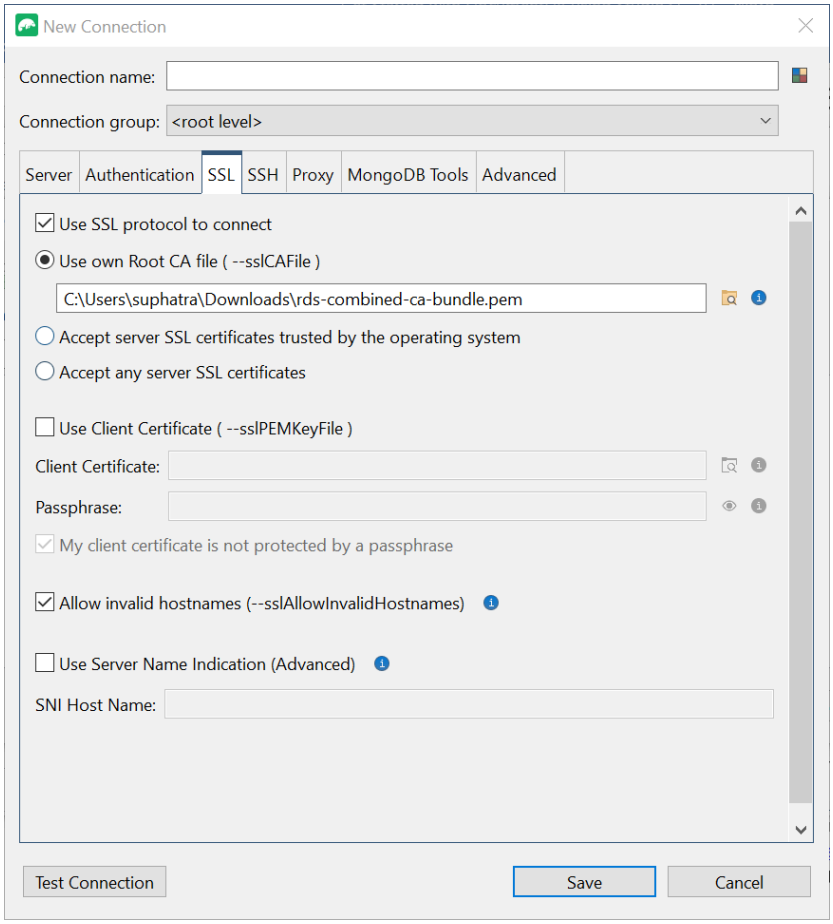


> **Note**
>
> Informationen zum Ermitteln Ihres Cluster-Endpunkts finden Sie unterDen Endpunkt einer Instanz finden.

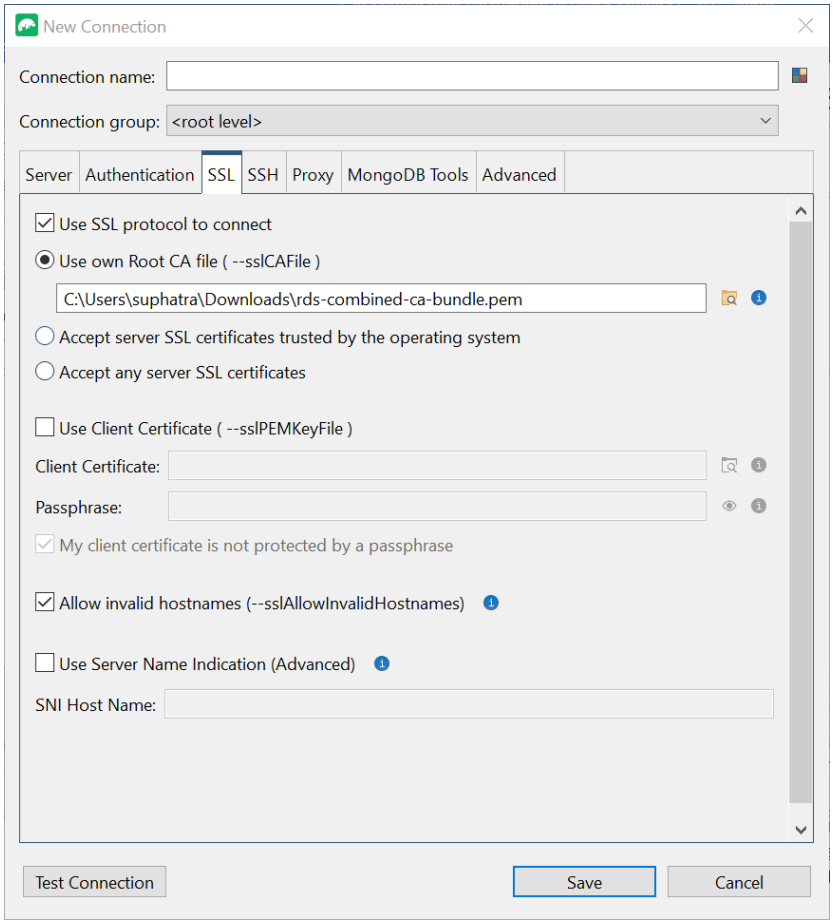
4. Wählen Sie die Registerkarte Authentifizierung und dann im Dropdownmenü für Authentifizierungsmodus die Option Legacy aus.



5. Geben Sie Ihren Benutzernamen und Ihre Anmeldeinformationen in die Felder Benutzername und Passwort ein.
6. Wählen Sie die Registerkarte SSL und aktivieren Sie das Kästchen SSL-Protokoll für die Verbindung verwenden.



7. Wählen Sie Eigene Root-CA-Datei verwenden. Fügen Sie dann das Amazon DocumentDB DocumentDB-Zertifikat hinzu (Sie können diesen Schritt überspringen, wenn SSL in Ihrem DocumentDB-Cluster deaktiviert ist). Markieren Sie das Kästchen, um ungültige Hostnamen zuzulassen.



> **Note**
>
> Sie haben das Zertifikat nicht? Sie können es mit dem folgenden Befehl herunterladen: `wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem`

8. Wenn Sie von einem Client-Computer außerhalb der Amazon VPC aus eine Verbindung herstellen, müssen Sie einen SSH-Tunnel erstellen. Sie werden dies auf der Registerkarte SSH tun.

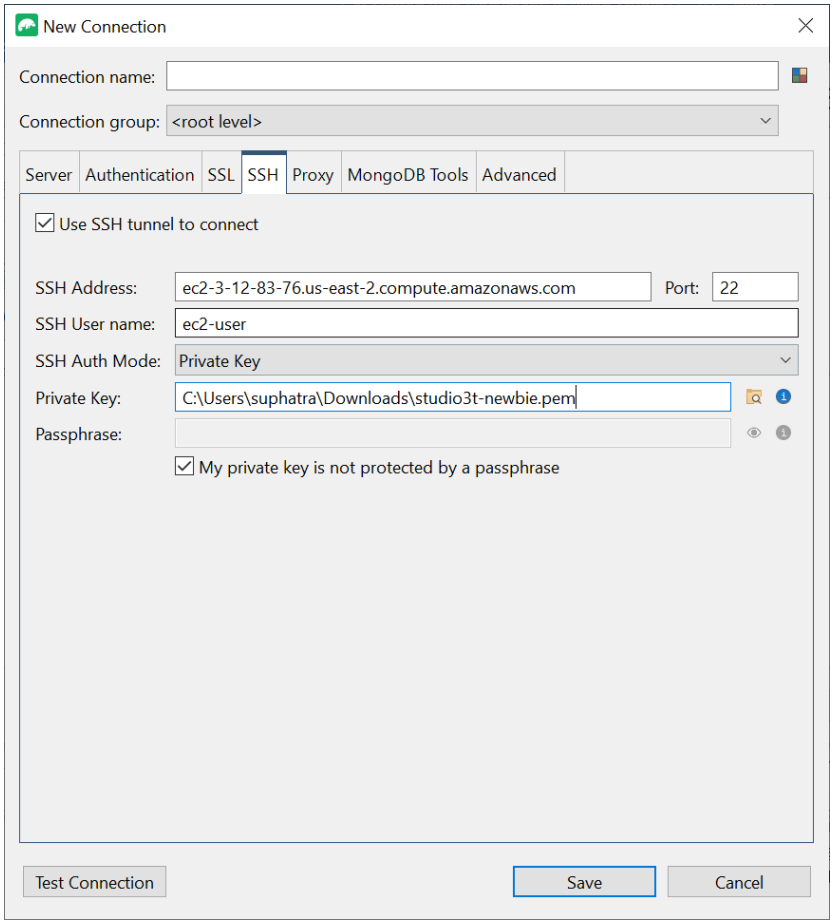
   a. Aktivieren Sie das Kontrollkästchen SSH-Tunnel verwenden und geben Sie die SSH-Adresse in das Feld SSH-Adresse ein. Dies ist Ihre Instanz Public DNS (IPV4). Sie können diese URL von Ihrer Amazon EC2-Managementkonsole abrufen.

   b. Geben Sie Ihren Nutzernamen ein. Dies ist der Benutzername Ihrer Amazon EC2 EC2-Instance

c. Wählen Sie für den SSH-Authentifizierungsmodus die Option Private Key aus. Wählen Sie im Feld Privater Schlüssel das Dateifinder-Symbol aus, um den privaten Schlüssel Ihrer Amazon EC2 EC2-Instance zu suchen und auszuwählen. Dies ist die PEM-Datei (key pair), die Sie beim Erstellen Ihrer Instance in der Amazon EC2 EC2-Konsole gespeichert haben.

d. Wenn Sie sich auf einem Linux/macOS Client-Computer befinden, müssen Sie möglicherweise die Berechtigungen Ihres privaten Schlüssels mit dem folgenden Befehl ändern:

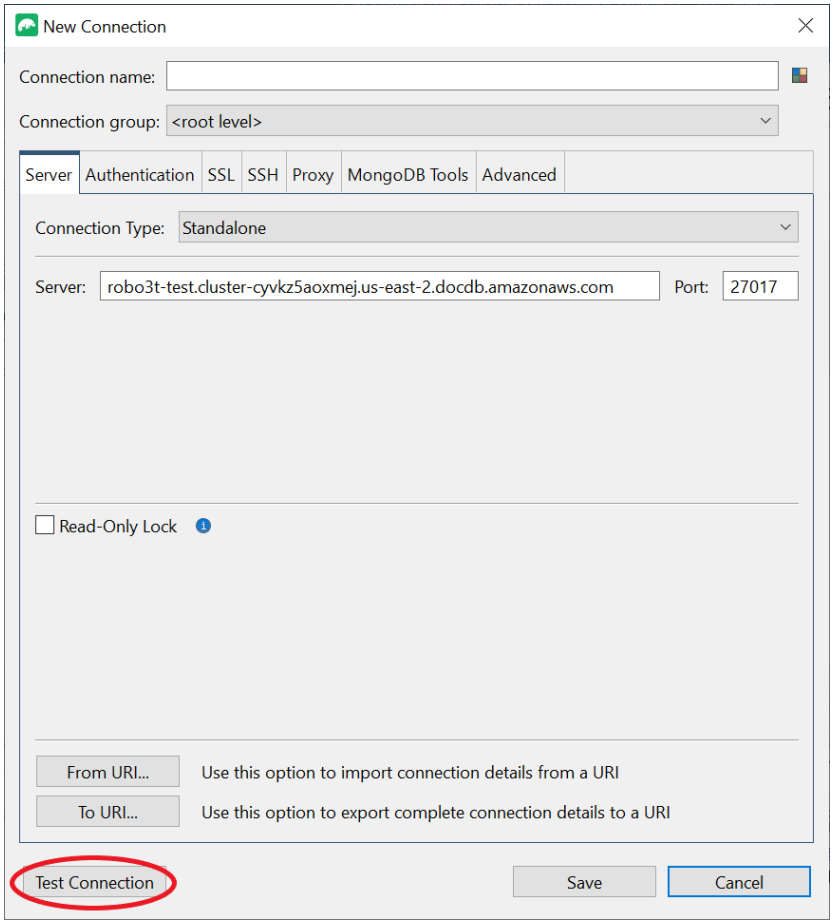
```
chmod 400 /fullPathToYourPemFile/<yourKey>.pem
```



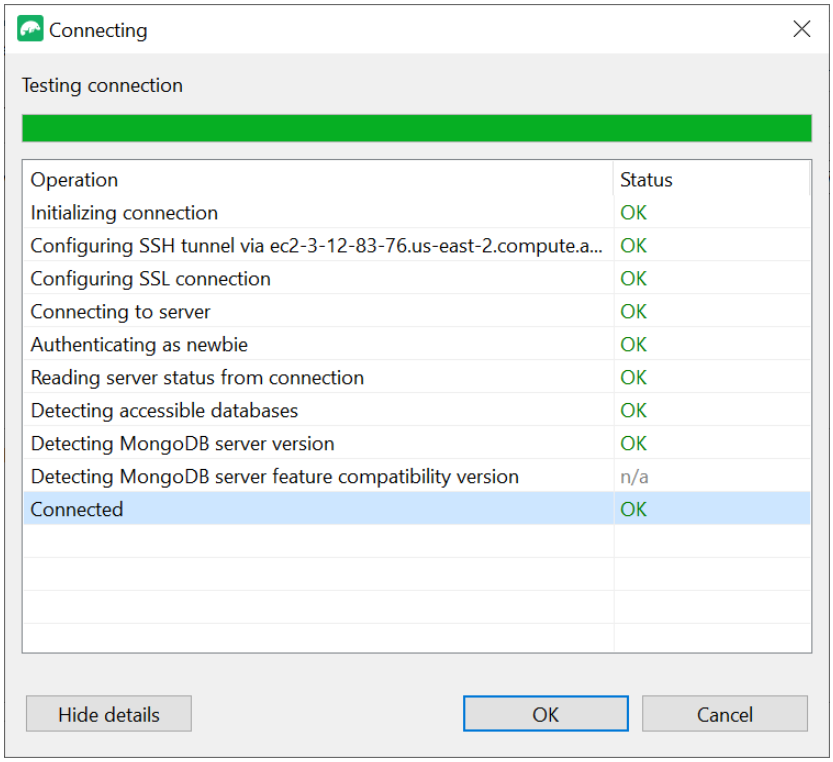
> **Note**
>
> Diese Amazon EC2 EC2-Instance sollte sich in derselben Amazon VPC und Sicherheitsgruppe wie Ihr DocumentDB-Cluster befinden. Sie können die SSH-

Adresse, den Benutzernamen und den privaten Schlüssel von Ihrer Amazon EC2-Managementkonsole abrufen.

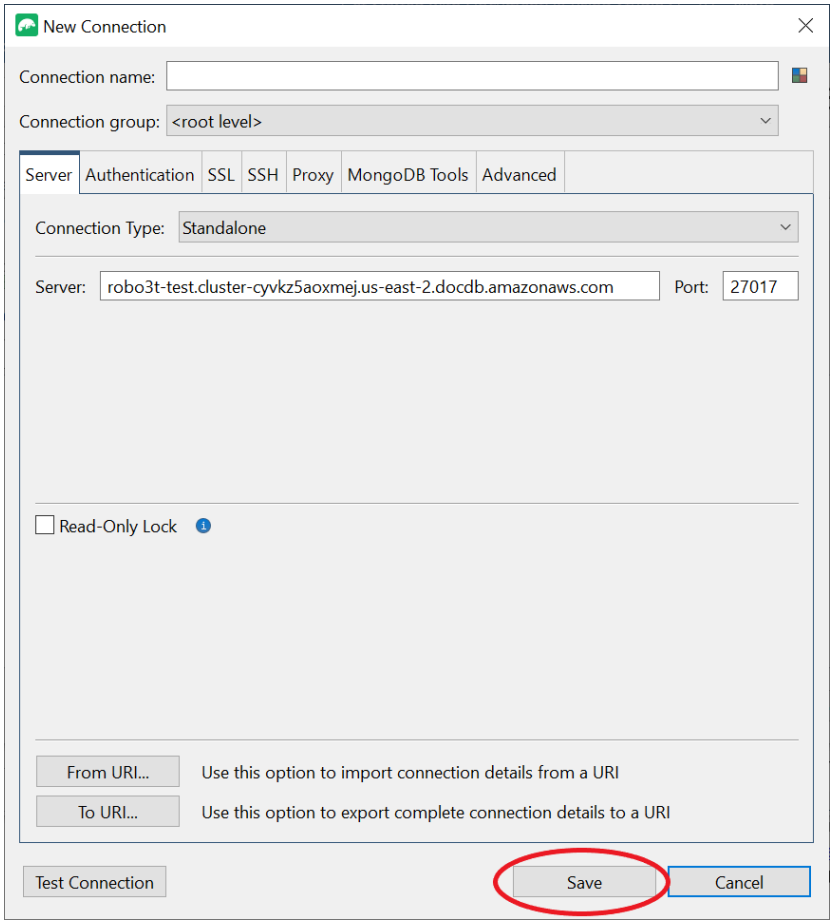
9. Testen Sie nun Ihre Konfiguration, indem Sie auf die Schaltfläche Verbindung testen klicken.



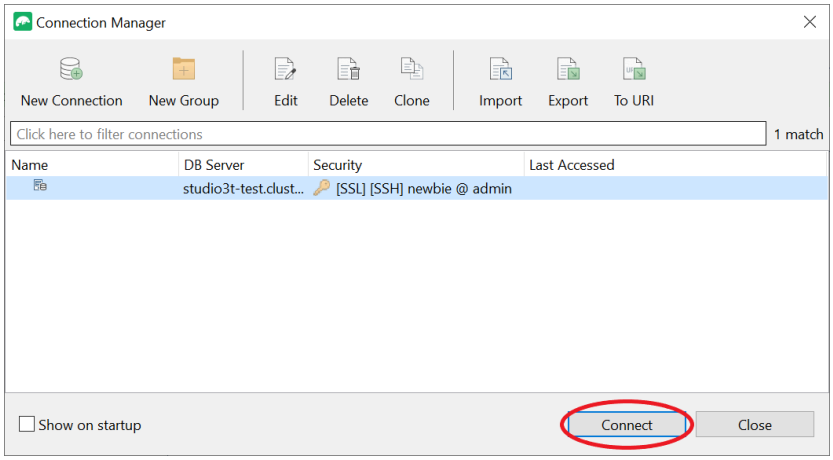
10. In einem Diagnosefenster sollte ein grüner Balken angezeigt werden, der anzeigt, dass der Test erfolgreich war. Wählen Sie nun OK, um das Diagnosefenster zu schließen.



11. Wählen Sie Speichern, um Ihre Verbindung für die future Verwendung zu speichern.



12. Wählen Sie nun Ihren Cluster aus und wählen Sie Connect.



Herzlichen Glückwunsch! Sie sind jetzt erfolgreich über Studio 3T mit Ihrem Amazon DocumentDB-Cluster verbunden.

Connect zu Amazon DocumentDB her mit DataGrip

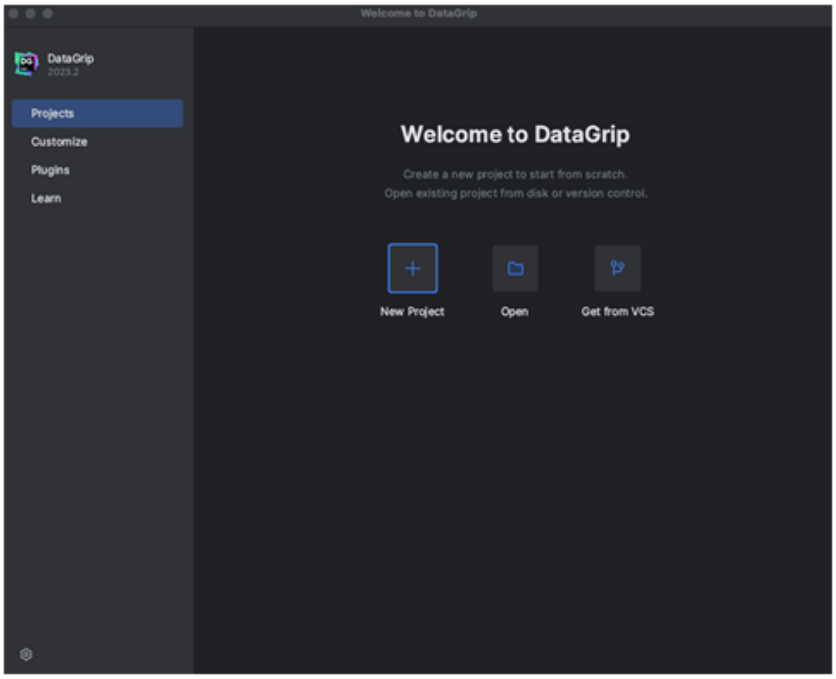
DataGripist eine leistungsstarke integrierte Entwicklungsumgebung (IDE), die verschiedene Datenbanksysteme unterstützt, darunter Amazon DocumentDB. Dieser Abschnitt führt Sie durch die Schritte, mit denen Sie eine Verbindung zu Ihrem Amazon DocumentDB-Cluster herstellen DataGrip, sodass Sie Ihre Daten einfach über eine grafische Oberfläche verwalten und abfragen können.

Voraussetzungen

- DataGrip IDE ist auf Ihrem Computer installiert. Sie können es von herunterladen JetBrains.
- Eine Amazon EC2 EC2-Instance, die in derselben VPC wie Ihr Amazon DocumentDB-Cluster läuft. Sie verwenden diese Instance, um einen sicheren Tunnel von Ihrem lokalen Computer zum Amazon DocumentDBCluster einzurichten. Folgen Sie den Anweisungen dazu. Stellen Sie mithilfe von Amazon EC2 eine Verbindung her
- Alternative zu einer Amazon EC2 EC2-Instance, einer VPN-Verbindung oder wenn Sie bereits über ein sicheres VPN auf Ihre AWS Infrastruktur zugreifen. Wenn Sie diese Option bevorzugen, folgen Sie den Anweisungen für den sicheren Zugriff auf Amazon DocumentDB mit AWS Client VPN.

Connect mit DataGrip

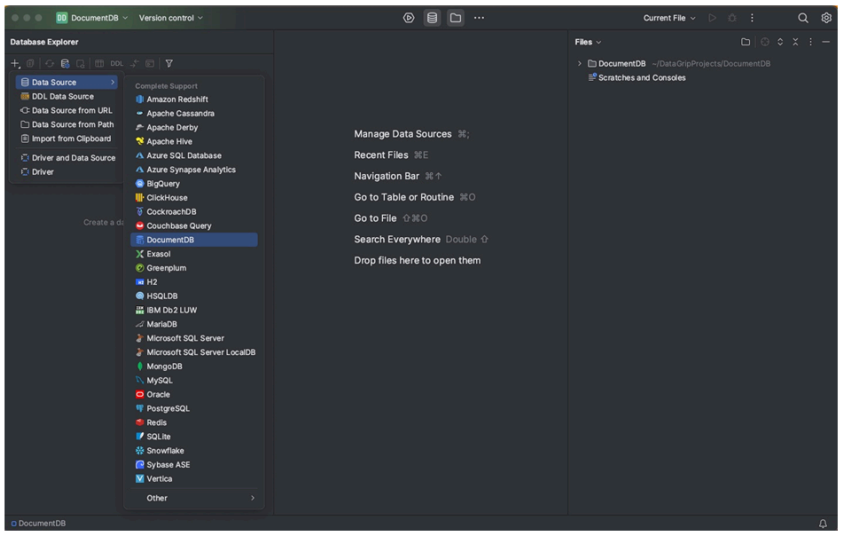
1. Starten Sie DataGrip auf Ihrem Computer und erstellen Sie ein neues Projekt.



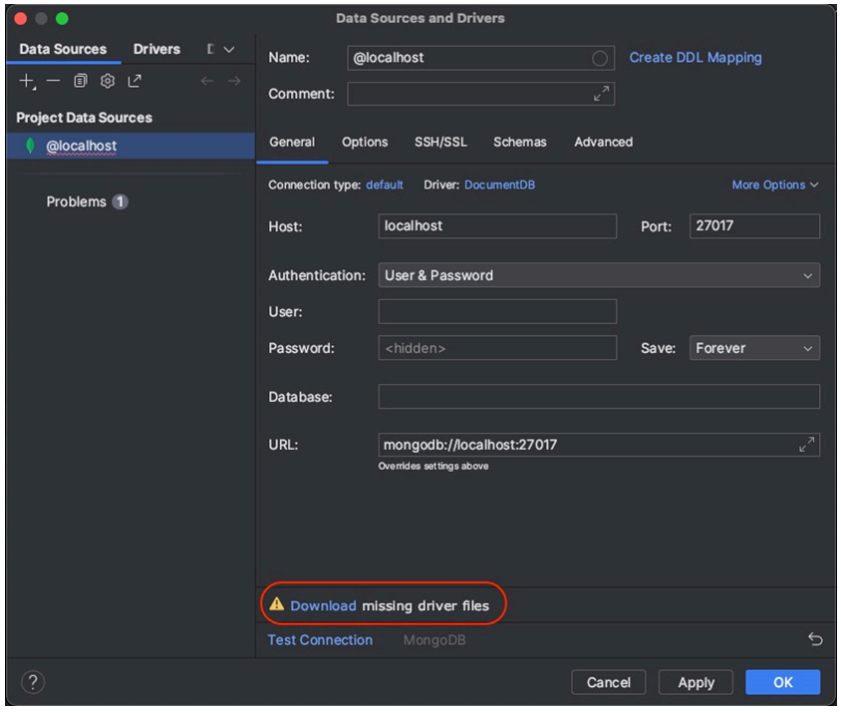
2. Fügen Sie mithilfe einer der folgenden Methoden eine neue Datenquelle hinzu:

a. Navigieren Sie im Hauptmenü zu Datei — Neu — Datenquelle und wählen Sie DocumentDB

b. Wählen Sie im Datenbank-Explorer das neue Symbol (+) in der Werkzeugleiste. Navigieren Sie zu Data Source und wählen Sie DocumentDB aus.



3. Überprüfen Sie auf der Seite Datenquellen auf der Registerkarte Allgemein, ob sich unten im Bereich mit den Verbindungseinstellungen der Link Fehlende Treiberdateien herunterladen befindet. Wählen Sie diesen Link, um Treiber herunterzuladen, die für die Interaktion mit einer Datenbank erforderlich sind. Einen direkten Download-Link finden Sie unter JetBrains JDBC-Treiber.



4. Geben Sie auf der Registerkarte Allgemein die Verbindungsdetails an:

a. Geben Sie im Feld Host den Amazon DocumentDB-Cluster-Endpunkt an.

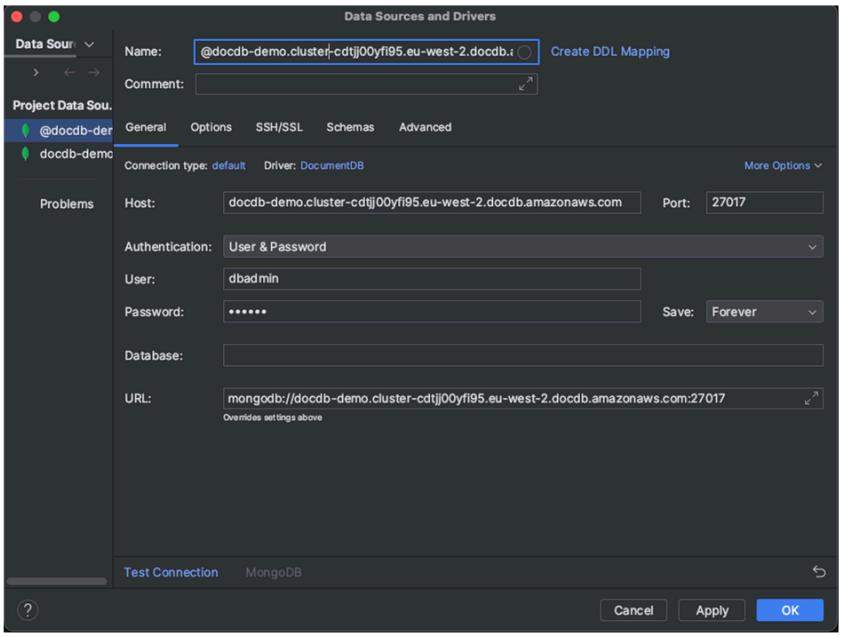
b. Der Port ist bereits auf 27017 eingestellt. Ändern Sie es, wenn Ihr Cluster auf einem anderen Port bereitgestellt wurde.

c. Wählen Sie für Authentifizierung die Option Benutzer und Passwort aus.

d. Geben Sie Ihren Benutzernamen und Ihr Passwort ein.

e. Das Datenbankfeld ist optional. Sie können die Datenbank angeben, zu der Sie eine Verbindung herstellen möchten.

f. Das URL-Feld wird automatisch vervollständigt, wenn Sie die vorherigen Details hinzufügen.



5. Aktivieren Sie auf der SSH/SSLRegisterkarte die Option SSH-Tunnel verwenden und wählen Sie dann das Symbol, um den SSH-Konfigurationsdialog zu öffnen. Geben Sie die folgenden Informationen ein:

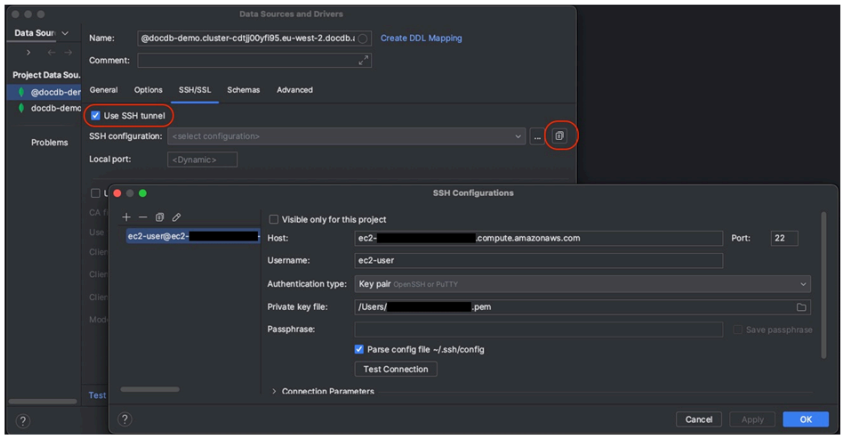
a. geben Sie im Feld Host den Hostnamen Ihrer Amazon EC2 EC2-Instance ein.

b. Geben Sie den Benutzernamen und das Passwort für Ihre Amazon EC2 EC2-Instance ein.

c. Wählen Sie als Authentifizierungstyp die Option Schlüsselpaar aus.

d. Geben Sie Ihre private Schlüsseldatei ein.

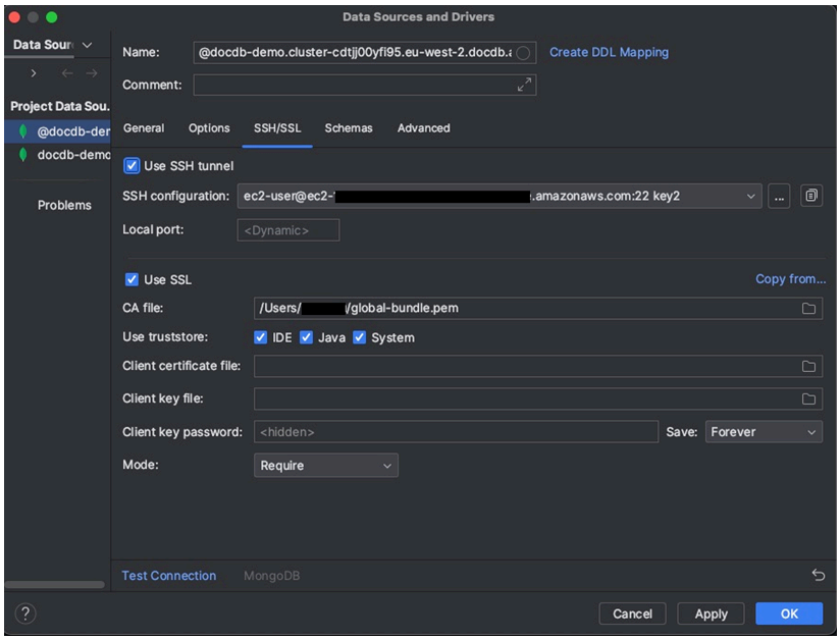
> **Note**
>
> Wenn Sie die VPN-Option verwenden, müssen Sie den SSH-Tunnel nicht konfigurieren.



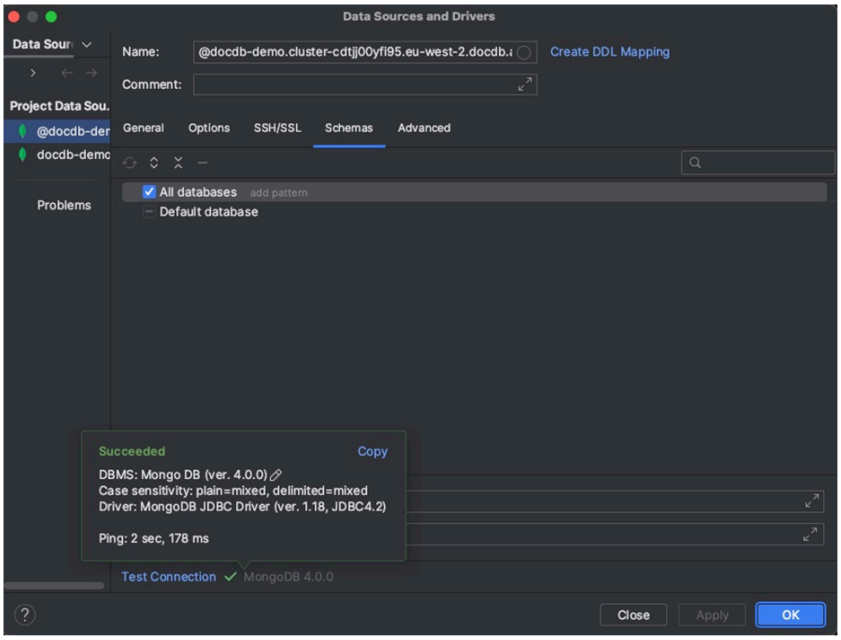
6. Aktivieren SSH/SSLSie auf der Registerkarte SSL verwenden. Geben Sie im Feld CA-Datei den Speicherort der `global-bundle.pem` Datei auf Ihrem Computer ein. Behalten Sie für Modus die Option Erforderlich bei.

> **Note**
>
> Sie können das Zertifikat von diesem Speicherort oder mit dem folgenden Befehl herunterladen: wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem

> **Note**
>
> Wenn Sie eine Verbindung zu Amazon DocumentDB Elastic Cluster herstellen, müssen Sie die CA-Datei nicht angeben. Lassen Sie die Option SSL verwenden aktiviert und lassen Sie alle anderen Optionen auf ihren Standardwerten stehen.



7. Wählen Sie auf der Registerkarte Schemas die Option Alle Datenbanken aus oder geben Sie den Filter „*: *“ in das Feld Schemamuster ein. Wählen Sie Verbindung testen, um die Verbindung zu testen.



8. Sobald die Verbindung erfolgreich getestet wurde, wählen Sie OK, um die Datenquellenkonfiguration zu speichern.

DataGrip features

DataGrip bietet verschiedene Funktionen, die Ihnen helfen, effizient mit Amazon DocumentDB zu arbeiten:

- SQL Editor — Schreiben Sie SQL-like Abfragen in Ihren DocumentDB-Sammlungen mit dem SQL-Editor in DataGrip und führen Sie sie aus.
- Visual Query Builder — Verwenden Sie den Visual Query Builder, um Abfragen grafisch zu erstellen, ohne SQL-Code schreiben zu müssen.
- Schemaverwaltung — Einfache Verwaltung Ihres Datenbankschemas, einschließlich des Erstellens, Änderns und Löschens von Sammlungen.
- Datenvisualisierung — Zeigen Sie Ihre Daten an und analysieren Sie sie mithilfe verschiedener Visualisierungstools, die unter verfügbar sind. DataGrip
- Daten exportieren und importieren — Übertragen Sie Daten zwischen Amazon DocumentDB und anderen Datenbanken mithilfe DataGrip der Export- und Importfunktionen.

Weiterführende Funktionen und Tipps zur Arbeit mit Amazon DocumentDB und anderen Datenbanksystemen finden Sie in der offiziellen DataGrip Dokumentation.

Stellen Sie mithilfe von Amazon EC2 eine Verbindung her

In diesem Abschnitt wird beschrieben, wie Sie die Konnektivität zwischen einem Amazon DocumentDB-Cluster und Amazon EC2 einrichten und von der Amazon EC2-Instance aus auf den Amazon DocumentDB-Cluster zugreifen.

Es gibt zwei Optionen für die Konfiguration der EC2-Verbindung:

- Automatisches Verbinden Ihrer EC2-Instance mit einer Amazon DocumentDB-Datenbank — Verwenden Sie die automatische Verbindungsfunktion in der EC2-Konsole, um die Verbindung zwischen Ihrer EC2-Instance und einer neuen oder vorhandenen Amazon DocumentDB-Datenbank automatisch zu konfigurieren. Diese Verbindung ermöglicht die Übertragung des Datenverkehrs zwischen der EC2-Instance und der Amazon DocumentDB-Datenbank. Diese Option wird in der Regel zum Testen und Erstellen neuer Sicherheitsgruppen verwendet.
- Manuelles Verbinden Ihrer EC2-Instance mit Ihrer Amazon DocumentDB-Datenbank — Konfigurieren Sie die Verbindung zwischen Ihrer EC2-Instance und Ihrer Amazon DocumentDB-Datenbank, indem Sie die Sicherheitsgruppen manuell konfigurieren und zuweisen, um die

Konfiguration zu reproduzieren, die durch die automatische Verbindungsfunktion erstellt wird. Diese Option wird in der Regel verwendet, um erweiterte Einstellungen zu ändern und vorhandene Sicherheitsgruppen zu verwenden.

Voraussetzungen

Unabhängig von der Option und bevor Sie Ihren ersten Amazon DocumentDB-Cluster erstellen, müssen Sie Folgendes tun:

Erstellen Sie ein Amazon Web Services (AWS) -Konto

Bevor Sie Amazon DocumentDB verwenden können, müssen Sie über ein Amazon Web Services (AWS) -Konto verfügen. Das AWS Konto ist kostenlos. Sie zahlen nur für die Services und Ressourcen, die Sie wirklich nutzen.

Wenn Sie noch keines haben AWS-Konto, führen Sie die folgenden Schritte aus, um eines zu erstellen.

Um sich für eine anzumelden AWS-Konto

1. Offen https://signin.aws.amazon.com/signup? request_type=registrieren.
2. Folgen Sie den Online-Anweisungen.

   Während der Anmeldung erhalten Sie einen Telefonanruf oder eine Textnachricht und müssen einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für eine registrieren AWS-Konto, wird eine erstellt. Root-Benutzer des AWS-Kontos Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Benutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

Richten Sie optional die erforderlichen AWS Identity and Access Management (IAM-) Berechtigungen ein.

Für den Zugriff auf die Verwaltung von Amazon DocumentDB-Ressourcen wie Clustern, Instances und Cluster-Parametergruppen sind Anmeldeinformationen erforderlich, mit denen Sie Ihre Anfragen authentifizieren AWS können. Weitere Informationen finden Sie unter Identitäts- und Zugriffsmanagement für Amazon DocumentDB.

1. Geben Sie in der Suchleiste von IAM ein **IAM** und wählen Sie im daraufhin angezeigten Drop-down-Menü die Option aus. AWS-Managementkonsole
2. Sobald Sie sich in der IAM-Konsole befinden, wählen Sie im Navigationsbereich Benutzer aus.
3. Wählen Sie Ihren Nutzernamen aus.
4. Wählen Sie Add permissions (Berechtigungen hinzufügen) aus.
5. Wählen Sie die Option Attach existing policies directly (Vorhandene Richtlinien direkt anfügen) aus.
6. Geben Sie `AmazonDocDBFullAccess` in die Suchleiste ein und wählen Sie ihn aus, sobald er in den Suchergebnissen erscheint.
7. Wählen Sie Weiter: Prüfen aus.
8. Wählen Sie Add permissions.

Erstellen Sie eine Amazon Virtual Private Cloud (Amazon VPC)

Je nachdem, in welcher Region AWS-Region Sie sich befinden, haben Sie möglicherweise bereits eine Standard-VPC erstellt. Wenn Sie keine Standard-VPC haben, führen Sie Schritt 1 der Anleitung Erste Schritte mit Amazon VPC im Amazon VPC-Benutzerhandbuch aus. Dies dauert weniger als fünf Minuten.

Stellen Sie automatisch eine Verbindung zu Amazon EC2 her

Topics

- Verbinden Sie automatisch eine EC2-Instance mit einer neuen Amazon DocumentDB-Datenbank
- Verbinden Sie eine EC2-Instance automatisch mit einer vorhandenen Amazon DocumentDB-Datenbank
- Übersicht über die automatische Verbindung mit einer EC2-Instance
- Anzeigen verbundener Rechenressourcen

Bevor Sie eine Verbindung zwischen einer EC2-Instance und einer neuen Amazon DocumentDB-Datenbank einrichten, stellen Sie sicher, dass Sie die unter beschriebenen Anforderungen erfüllen. Übersicht über die automatische Verbindung mit einer EC2-Instance Wenn Sie nach der Konfiguration der Konnektivität Änderungen an Sicherheitsgruppen vornehmen, können sich

die Änderungen auf die Verbindung zwischen der EC2-Instance und der Amazon DocumentDB-Datenbank auswirken.

> **Note**
>
> Sie können nur automatisch eine Verbindung zwischen einer EC2-Instance und einer Amazon DocumentDB-Datenbank einrichten, indem Sie die verwenden. AWS-Managementkonsole Sie können keine automatische Verbindung mit der AWS CLI oder der Amazon DocumentDB-API einrichten.

Verbinden Sie automatisch eine EC2-Instance mit einer neuen Amazon DocumentDB-Datenbank

Im folgenden Prozess wird davon ausgegangen, dass Sie die Schritte im Thema abgeschlossen haben. Voraussetzungen

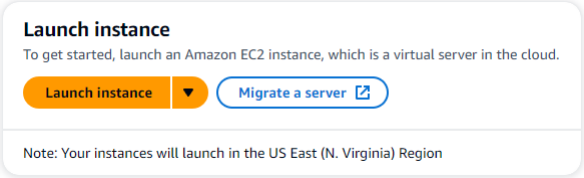
Schritte

- Schritt 1: Erstellen Sie eine Amazon EC2-Instance
- Schritt 2: Erstellen Sie einen Amazon DocumentDB-Cluster
- Schritt 3: Stellen Sie eine Verbindung zu Ihrer Amazon EC2-Instance her
- Schritt 4: Installieren Sie die MongoDB Shell
- Schritt 5: Amazon DocumentDB TLS verwalten
- Schritt 6: Stellen Sie eine Verbindung zu Ihrem Amazon DocumentDB-Cluster her
- Schritt 7: Daten einfügen und abfragen
- Schritt 8: Erkunden

Schritt 1: Erstellen Sie eine Amazon EC2-Instance

In diesem Schritt erstellen Sie eine Amazon EC2-Instance in derselben Region und Amazon VPC, die Sie später zur Bereitstellung Ihres Amazon DocumentDB-Clusters verwenden werden.

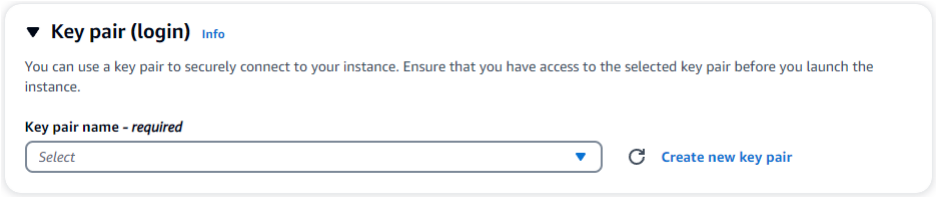
1. Wählen Sie auf der Amazon EC2-Konsole Launch Instance aus.



2. Geben Sie einen Namen oder eine Kennung in das Feld Name ein, das sich im Abschnitt Name und Tags befindet.
3. Suchen Sie in der Dropdownliste Amazon Machine Image (AMI) nach Amazon Linux 2 AMI und wählen Sie es aus.

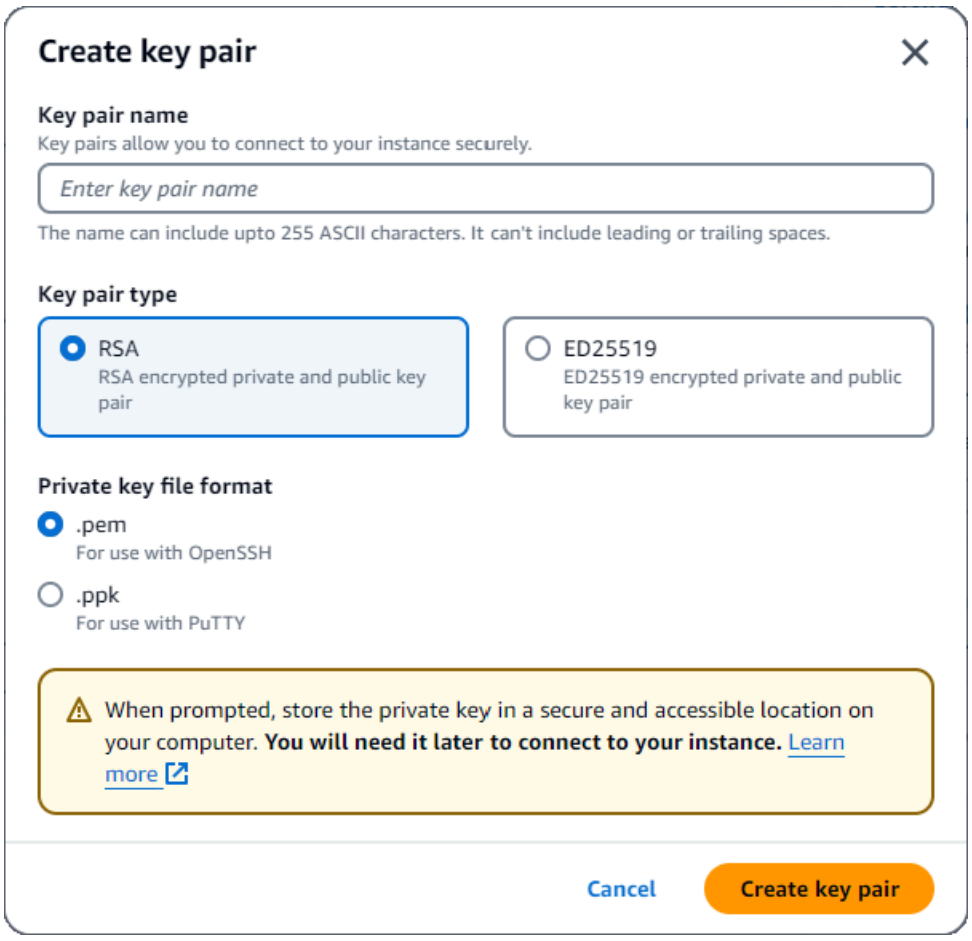


4. Suchen Sie in der Dropdownliste Instance-Typ nach t3.micro und wählen Sie es aus.
5. Geben Sie im Abschnitt Schlüsselpaar (Anmeldung) die ID eines vorhandenen Schlüsselpaars ein, oder wählen Sie Neues Schlüsselpaar erstellen.



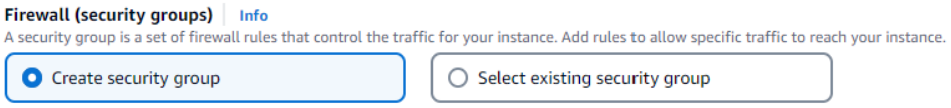
Sie müssen ein Amazon EC2-Schlüsselpaar angeben.

- Wenn Sie ein Amazon EC2-Schlüsselpaar haben:
  1. Wählen Sie ein Schlüsselpaar aus, wählen Sie Ihr Schlüsselpaar aus der Liste aus.
  2. Sie müssen bereits über die private Schlüsseldatei (.pem- oder .ppk-Datei) verfügen, um sich bei Ihrer Amazon EC2-Instance anmelden zu können.
- Wenn Sie kein Amazon EC2-Schlüsselpaar haben:
  1. Wählen Sie Neues Schlüsselpaar erstellen. Das Dialogfeld Schlüsselpaar erstellen wird angezeigt.
  2. Geben Sie einen Namen in das Feld Schlüsselpaar-Name ein.
  3. Wählen Sie den Schlüsselpaartyp und das Dateiformat für den privaten Schlüssel.
  4. Wählen Sie Create Key Pair (Schlüsselpaar erstellen) aus.



> **Note**
>
> Aus Sicherheitsgründen empfehlen wir dringend, ein Schlüsselpaar sowohl für die SSH- als auch für die Internetverbindung zu Ihrer EC2-Instance zu verwenden.

6. Optional: Wählen Sie im Abschnitt Netzwerkeinstellungen unter Firewall (Sicherheitsgruppen) die Option Sicherheitsgruppe erstellen aus.



Wählen Sie Sicherheitsgruppe erstellen (überprüfen Sie alle Regeln, die den Datenverkehr zulassen, die für Ihre EC2-Konnektivität gelten).

> **Note**
>
> Wenn Sie eine vorhandene Sicherheitsgruppe verwenden möchten, folgen Sie den Anweisungen unterStellen Sie eine manuelle Verbindung zu Amazon EC2 her.

7. Überprüfen Sie im Abschnitt Zusammenfassung Ihre EC2-Konfiguration und wählen Sie Launch Instance, falls dies korrekt ist.

Schritt 2: Erstellen Sie einen Amazon DocumentDB-Cluster

Während die Amazon EC2-Instance bereitgestellt wird, erstellen Sie Ihren Amazon DocumentDB-Cluster.

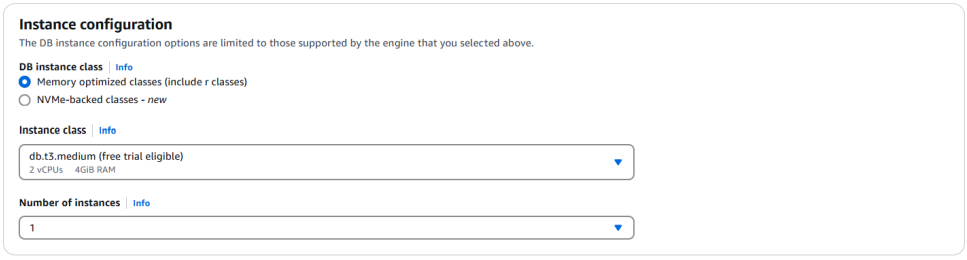
1. Navigieren Sie zur Amazon DocumentDB-Konsole und wählen Sie im Navigationsbereich Clusters aus.
2. Wählen Sie Erstellen aus.
3. Belassen Sie die Einstellung für den Clustertyp auf der Standardeinstellung Instance Based Cluster.
4. Geben Sie in der Cluster-Konfiguration als Cluster-ID einen eindeutigen Namen ein. Beachten Sie, dass die Konsole alle Clusternamen in Kleinbuchstaben umwandelt, unabhängig davon, wie sie eingegeben werden.

   Belassen Sie die Engine-Version auf dem Standardwert 5.0.0.
5. Belassen Sie für die Cluster-Speicherkonfiguration die Standardeinstellung von Amazon DocumentDB Standard.
6. In der Instanzkonfiguration:

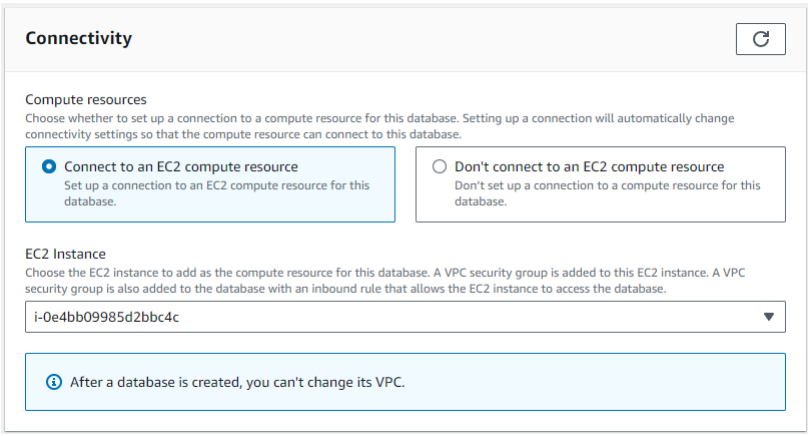
   - Wählen Sie für die DB-Instance-Klasse die Option Speicheroptimierte Klassen (einschließlich R-Klassen) aus (dies ist die Standardeinstellung).

     Die andere Instanzoption sind NVMe-backed Klassen. Weitere Informationen hierzu finden Sie unter NVMe-backed Instanzen.
   - Wählen Sie für die Instance-Klasse den Instanztyp aus, der Ihren Anforderungen entspricht. Eine detailliertere Erläuterung der Instanzklassen finden Sie unterSpezifikationen der Instanzklasse.

- Wählen Sie für die Anzahl der Instances eine Zahl aus, die Ihren Anforderungen am besten entspricht. Denken Sie daran, je niedriger die Anzahl, desto niedriger sind die Kosten und desto geringer ist das read/write Volumen, das vom Cluster verwaltet werden kann.



7. Wählen Sie für Konnektivität die Option Mit einer EC2-Rechenressource verbinden aus. Dies ist die EC2-Instance, die Sie in Schritt 1 erstellt haben.



> **Note**
>
> Wenn Sie eine Verbindung zu einer EC2-Rechenressource herstellen, wird automatisch eine Sicherheitsgruppe für Ihre EC2-Rechenressourcenverbindung zu Ihrem Amazon DocumentDB-Cluster erstellt. Wenn Sie die Erstellung Ihres Clusters abgeschlossen haben und die neu erstellte Sicherheitsgruppe sehen möchten, navigieren Sie zur Cluster-Liste und wählen Sie die ID Ihres Clusters aus. Gehen Sie auf der Registerkarte Konnektivität und Sicherheit zu Sicherheitsgruppen und suchen Sie Ihre Gruppe unter Sicherheitsgruppenname (ID). Es wird den Namen Ihres Clusters enthalten und ungefähr so aussehen:`docdb-ec2-docdb-2023-12-11-21-33-41:i-0e4bb09985d2bbc4c (sg-0238e0b0bf0f73877)`.

8. Geben Sie im Abschnitt Authentifizierung einen Benutzernamen für den Hauptbenutzer ein und wählen Sie dann Selbstverwaltet aus. Geben Sie ein Passwort ein und bestätigen Sie es.

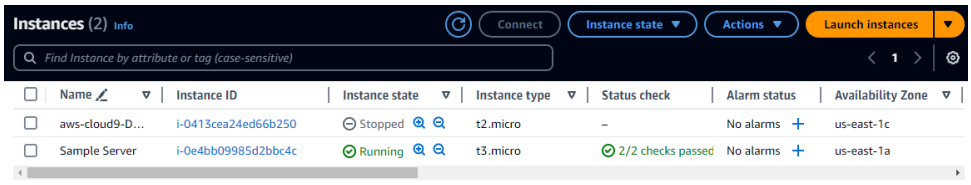
Wenn Sie stattdessen Verwaltet in ausgewählt haben AWS Secrets Manager, finden Sie Passwortverwaltung mit Amazon DocumentDB und AWS Secrets Manager weitere Informationen unter.

9. Wählen Sie Cluster erstellen.

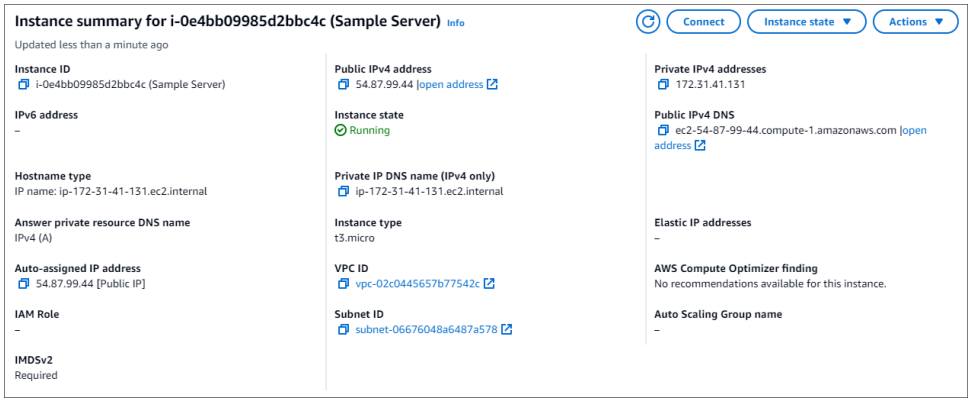
Schritt 3: Stellen Sie eine Verbindung zu Ihrer Amazon EC2-Instance her

Um die Mongo-Shell zu installieren, müssen Sie zuerst eine Verbindung zu Ihrer Amazon EC2-Instance herstellen. Durch die Installation der Mongo-Shell können Sie eine Verbindung zu Ihrem Amazon DocumentDB-Cluster herstellen und diesen abfragen. Führen Sie folgende Schritte aus:

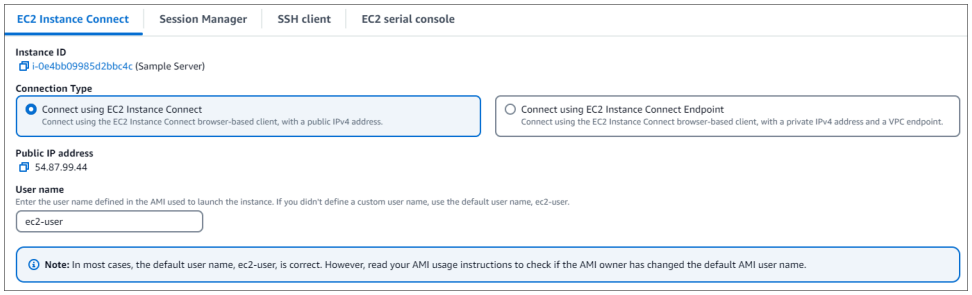
1. Navigieren Sie auf der Amazon EC2-Konsole zu Ihren Instances und prüfen Sie, ob die Instance, die Sie gerade erstellt haben, läuft. Wenn ja, wählen Sie die Instance aus, indem Sie auf die Instance-ID klicken.



2. Wählen Sie Connect aus.



3. Es gibt vier Optionen mit Registerkarten für Ihre Verbindungsmethode: Amazon EC2 Instance Connect, Session Manager, SSH-Client oder serielle EC2-Konsole. Sie müssen eine auswählen und den Anweisungen folgen. Wenn Sie fertig sind, wählen Sie Verbinden.



> **Note**
>
> Wenn sich Ihre IP-Adresse geändert hat, nachdem Sie mit diesem Walkthrough begonnen haben, oder wenn Sie zu einem späteren Zeitpunkt zu Ihrer Umgebung zurückkehren, müssen Sie Ihre `demoEC2` Sicherheitsgruppenregel für eingehenden Datenverkehr aktualisieren, um eingehenden Datenverkehr von Ihrer neuen API-Adresse zu ermöglichen.

Schritt 4: Installieren Sie die MongoDB Shell

Sie können jetzt die MongoDB-Shell installieren, ein Befehlszeilenprogramm, mit dem Sie Ihren Amazon DocumentDB-Cluster verbinden und abfragen. Derzeit gibt es zwei Versionen der MongoDB-Shell: die neueste Version, Mongosh, und die vorherige Version, Mongo-Shell.

> **Important**
>
> Es gibt eine bekannte Einschränkung bei Node.js Treibern, die älter als Version 6.13.1 sind und die derzeit von der IAM-Identitätsauthentifizierung für Amazon DocumentDB nicht unterstützt werden. Node.js Treiber und Tools, die Node.js Treiber verwenden (z. B. Mongosh), müssen aktualisiert werden, um die Node.js Treiberversion 6.13.1 oder höher zu verwenden.

Folgen Sie den nachstehenden Anweisungen, um die MongoDB-Shell für Ihr Betriebssystem zu installieren.

On Amazon Linux

Um die MongoDB-Shell auf Amazon Linux zu installieren

Wenn Sie keine IAM-Authentifizierung verwenden und die neueste MongoDB-Shell (mongosh) verwenden möchten, um eine Verbindung zu Ihrem Amazon DocumentDB-Cluster herzustellen, gehen Sie wie folgt vor:

1. Erstellen Sie die Repository-Datei. Führen Sie in der Befehlszeile Ihrer EC2-Instance, die Sie erstellt haben, den folgenden Befehl aus:

```
echo -e "[mongodb-org-5.0] \nname=MongoDB Repository\nbaseurl=https://
repo.mongodb.org/yum/amazon/2023/mongodb-org/5.0/x86_64/\ngpgcheck=1 \nenabled=1
 \ngpgkey=https://pgp.mongodb.com/server-5.0.asc" | sudo tee /etc/yum.repos.d/
mongodb-org-5.0.repo
```

2. Wenn der Vorgang abgeschlossen ist, installieren Sie mongosh mit einer der beiden folgenden Befehlsoptionen an der Befehlszeile:

   Option 1 — Wenn Sie während der Amazon EC2-Konfiguration das standardmäßige Amazon Linux 2023 ausgewählt haben, geben Sie diesen Befehl ein:

```
sudo yum install -y mongodb-mongosh-shared-openssl3
```

   Option 2 — Wenn Sie während der Amazon EC2-Konfiguration Amazon Linux 2 ausgewählt haben, geben Sie diesen Befehl ein:

```
sudo yum install -y mongodb-mongosh
```

Wenn Sie die IAM-Authentifizierung verwenden, müssen Sie die vorherige Version der MongoDB-Shell (5.0) verwenden, um eine Verbindung zu Ihrem Amazon DocumentDB-Cluster herzustellen. Gehen Sie wie folgt vor:

1. Erstellen Sie die Repository-Datei. Führen Sie in der Befehlszeile Ihrer EC2-Instance, die Sie erstellt haben, den folgenden Befehl aus:

```
echo -e "[mongodb-org-5.0] \nname=MongoDB Repository\nbaseurl=https://
repo.mongodb.org/yum/amazon/2023/mongodb-org/5.0/x86_64/\ngpgcheck=1 \nenabled=1
 \ngpgkey=https://pgp.mongodb.com/server-5.0.asc" | sudo tee /etc/yum.repos.d/
mongodb-org-5.0.repo
```

2. Wenn der Vorgang abgeschlossen ist, installieren Sie die Mongodb 5.0-Shell mit der folgenden Befehlsoption an der Eingabeaufforderung:

```
sudo yum install -y mongodb-org-shell
```

On Ubuntu

Um mongosh auf Ubuntu zu installieren

1. Importieren Sie den öffentlichen Schlüssel, der von dem Paketverwaltungssystem verwendet wird.

   ```
   curl -fsSL https://pgp.mongodb.com/server-5.0.asc | sudo gpg --dearmor -o /usr/
   share/keyrings/mongodb-server-5.0.gpg
   ```

2. Erstellen Sie die Listendatei `mongodb-org-5.0.list` für MongoDB mit dem korrekten Befehl für Ihre Ubuntu-Version.

   ```
   echo "deb [ arch=amd64,arm64 signed-by=/usr/share/keyrings/mongodb-
   server-5.0.gpg ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/5.0
    multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-5.0.list
   ```

3. Importieren und aktualisieren Sie die lokale Paketdatenbank mit dem folgenden Befehl:

   ```
   sudo apt-get update
   ```

4. Installieren Sie Mongosh.

   ```
   sudo apt-get install -y mongodb-mongosh
   ```

Weitere Informationen zum Installieren von früheren Versionen von MongoDB auf Ihrem Ubuntu-System finden Sie unter Installieren von MongoDB Community Edition auf Ubuntu.

On other operating systems

Informationen zum Installieren der mongo-Shell auf anderen Betriebssystemen finden Sie unter Installieren von MongoDB Community Edition in der MongoDB-Dokumentation.

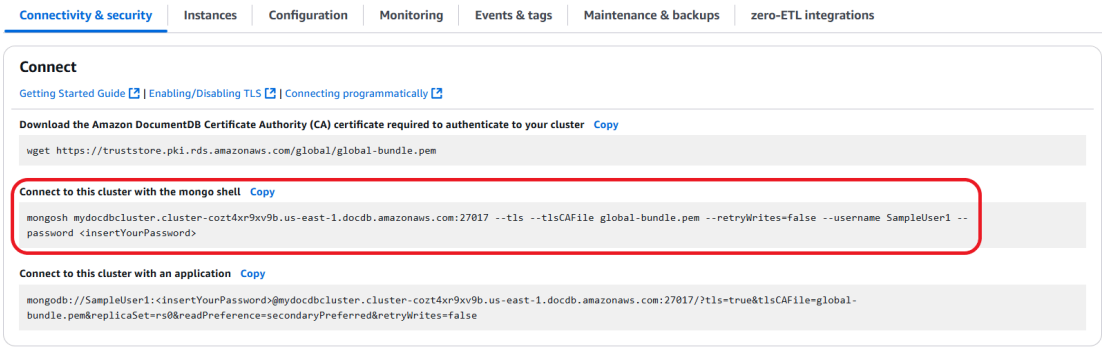
Schritt 5: Amazon DocumentDB TLS verwalten

Laden Sie das CA-Zertifikat für Amazon DocumentDB mit dem folgenden Code herunter: `wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem`

> **Note**
>
> Transport Layer Security (TLS) ist standardmäßig für alle neuen Amazon DocumentDB-Cluster aktiviert. Weitere Informationen finden Sie unter Verschlüsseln von Daten während der Übertragung.

Schritt 6: Stellen Sie eine Verbindung zu Ihrem Amazon DocumentDB-Cluster her

1. Suchen Sie auf der Amazon DocumentDB-Konsole unter Clusters nach Ihrem Cluster. Wählen Sie den Cluster aus, den Sie erstellt haben, indem Sie auf die Cluster-ID für diesen Cluster klicken.
2. Suchen Sie auf der Registerkarte Konnektivität und Sicherheit im Feld Verbinden nach Mit diesem Cluster verbinden mit der Mongo-Shell:



   Kopieren Sie die bereitgestellte Verbindungszeichenfolge und fügen Sie sie in Ihr Terminal ein.

   Nehmen Sie die folgenden Änderungen daran vor:

   a. Stellen Sie sicher, dass Sie den richtigen Benutzernamen in der Zeichenfolge haben.

   b. Lassen Sie es weg, `<insertYourPassword>` damit Sie bei der Verbindung von der Mongo-Shell nach dem Passwort gefragt werden.

   c. Optional: Wenn Sie die IAM-Authentifizierung verwenden oder die vorherige Version der MongoDB-Shell verwenden, ändern Sie Ihre Verbindungszeichenfolge wie folgt:

```
mongo --ssl --host mydocdbcluster.cluster-cozt4xr9xv9b.us-
east-1.docdb.amazonaws.com:27017 --sslCAFile global-bundle.pem --
username SampleUser1 --password
```

`mydocdbcluster.cluster-cozt4xr9xv9b.us-east-1`Ersetzen Sie es durch dieselben Informationen aus Ihrem Cluster.

3. Drücken Sie in Ihrem Terminal die Eingabetaste. Sie werden jetzt nach Ihrem Passwort gefragt. Geben Sie Ihr Passwort ein.
4. Wenn Sie Ihr Passwort eingeben und die `rs0 [direct: primary] <env-name>>` Aufforderung sehen, sind Sie erfolgreich mit Ihrem Amazon DocumentDB-Cluster verbunden.

Informationen zur Lösung von Verbindungsproblemen finden Sie unterProblembehandlung bei Amazon DocumentDB.

Schritt 7: Daten einfügen und abfragen

Jetzt, da Sie mit Ihrem Cluster verbunden sind, können Sie einige Abfragen ausführen, um sich mit der Verwendung einer Dokumentdatenbank vertraut zu machen.

1. Um ein einzelnes Dokument einzufügen, geben Sie Folgendes ein:

```
db.collection.insertOne({"hello":"DocumentDB"})
```

Sie erhalten die folgende Ausgabe:

```
{
  acknowledged: true,
  insertedId: ObjectId('673657216bdf6258466b128c')
}
```

2. Sie können das Dokument lesen, das Sie mit dem `findOne()` Befehl geschrieben haben (da nur ein einziges Dokument zurückgegeben wird). Geben Sie Folgendes ein:

```
db.collection.findOne()
```

Sie erhalten die folgende Ausgabe:

```
{ "_id" : ObjectId("5e401fe56056fda7321fbd67"), "hello" : "DocumentDB" }
```

3. Um ein paar weitere Abfragen durchzuführen, ziehen Sie einen Anwendungsfall mit Spielprofilen in Betracht. Fügen Sie zunächst einige Einträge in eine Sammlung mit dem Titel ein`profiles`. Geben Sie Folgendes ein:

```
db.profiles.insertMany([{ _id: 1, name: 'Matt', status: 'active', level: 12, score:
 202 },
     { _id: 2, name: 'Frank', status: 'inactive', level: 2, score: 9 },
     { _id: 3, name: 'Karen', status: 'active', level: 7, score: 87 },
     { _id: 4, name: 'Katie', status: 'active', level: 3, score: 27 }
])
```

   Sie erhalten die folgende Ausgabe:

```
{ acknowledged: true, insertedIds: { '0': 1, '1': 2, '2': 3, '3': 4 } }
```

4. Verwenden Sie den `find()` Befehl, um alle Dokumente in der Profilsammlung zurückzugeben. Geben Sie Folgendes ein:

```
db.profiles.find()
```

   Sie erhalten eine Ausgabe, die den Daten entspricht, die Sie in Schritt 3 eingegeben haben.

5. Verwenden Sie mithilfe eines Filters eine Abfrage für ein einzelnes Dokument. Geben Sie Folgendes ein:

```
db.profiles.find({name: "Katie"})
```

   Sie erhalten die folgende Ausgabe:

```
{ "_id" : 4, "name" : "Katie", "status": "active", "level": 3, "score":27}
```

6. Versuchen wir nun, ein Profil zu finden und es mit dem `findAndModify` Befehl zu ändern. Wir geben dem Benutzer Matt zusätzliche 10 Punkte mit dem folgenden Code:

```
db.profiles.findAndModify({
        query: { name: "Matt", status: "active"},
        update: { $inc: { score: 10 } }
```

```
    })
```

Du erhältst die folgende Ausgabe (beachte, dass seine Punktzahl noch nicht gestiegen ist):

```
{
    [{_id : 1, name : 'Matt', status: 'active', level: 12, score: 202}]
```

7. Mit der folgenden Abfrage können Sie überprüfen, ob sich sein Ergebnis geändert hat:

   ```
   db.profiles.find({name: "Matt"})
   ```

   Sie erhalten die folgende Ausgabe:

   ```
   { "_id" : 1, "name" : "Matt", "status" : "active", "level" : 12, "score" : 212 }
   ```

Schritt 8: Erkunden

Herzlichen Glückwunsch! Sie haben die Schnellstartanleitung für Amazon DocumentDB erfolgreich abgeschlossen.

Was kommt als Nächstes? Erfahren Sie, wie Sie diese leistungsstarke Datenbank mit einigen ihrer beliebten Funktionen voll ausschöpfen können:

- Verwaltung von Amazon DocumentDB-Clustern und -Instances
- Skalierung
- Sichern und Wiederherstellen

> **Note**
>
> Um Kosten zu sparen, können Sie entweder Ihren Amazon DocumentDB-Cluster stoppen, um die Kosten zu senken, oder den Cluster löschen. Standardmäßig stoppt Ihre AWS Cloud9 Umgebung nach 30 Minuten Inaktivität die zugrunde liegende Amazon EC2-Instance.

Verbinden Sie eine EC2-Instance automatisch mit einer vorhandenen Amazon DocumentDB-Datenbank

Das folgende Verfahren setzt voraus, dass Sie über einen vorhandenen Amazon DocumentDB-Cluster und eine bestehende Amazon EC2-Instance verfügen.

Greifen Sie auf Ihren Amazon DocumentDB-Cluster zu und richten Sie die Amazon EC2-Verbindung ein

1. Greifen Sie auf Ihren Amazon DocumentDB-Cluster zu.

   a. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb

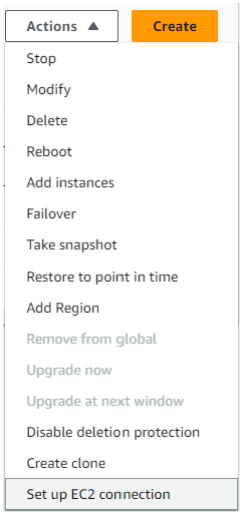
   b. Klicken Sie im Navigationsbereich auf Cluster.

   > **Tip**
   >
   > Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

   c. Geben Sie den gewünschten Cluster an, indem Sie auf die Schaltfläche links neben dem Clusternamen klicken.

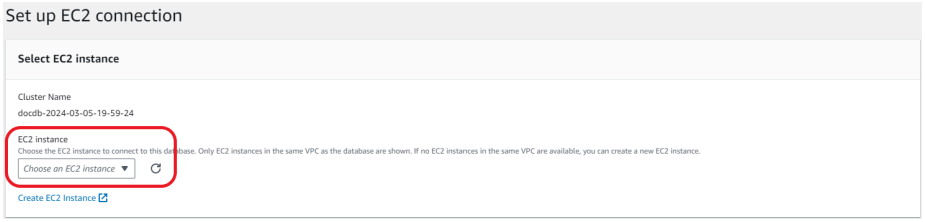
2. Richten Sie die Amazon EC2-Verbindung ein.

   a. Wählen Sie Aktionen und dann EC2-Verbindung einrichten aus.



   Das Dialogfeld EC2-Verbindung einrichten wird angezeigt.

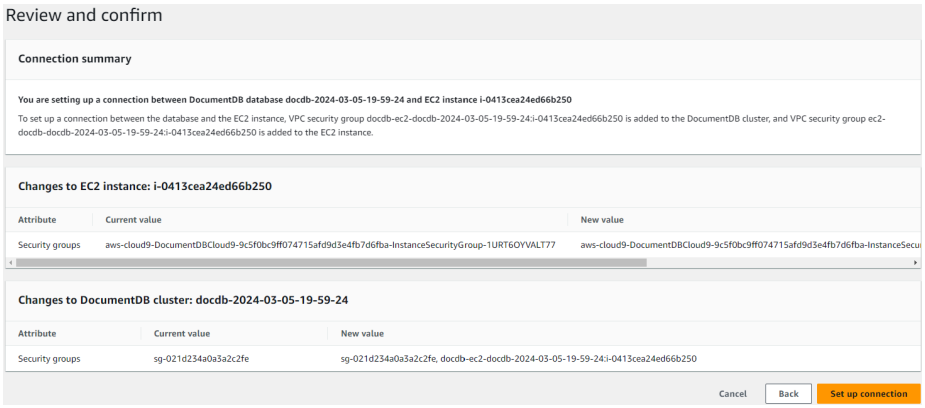
b. Wählen Sie im Feld EC2-Instance die EC2-Instance aus, die Sie mit Ihrem Cluster verbinden möchten.



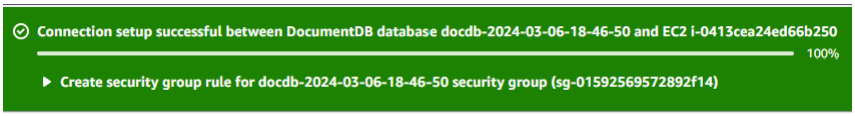
c. Klicken Sie auf Weiter.

Das Dialogfeld Überprüfen und Bestätigen wird angezeigt.

d. Vergewissern Sie sich, dass die Änderungen korrekt sind. Wählen Sie dann Verbindung einrichten.



Bei Erfolg wird die folgende Überprüfung angezeigt:



Übersicht über die automatische Verbindung mit einer EC2-Instance

Wenn Sie eine Verbindung zwischen einer EC2-Instance und einer Amazon DocumentDB-Datenbank einrichten, konfiguriert Amazon DocumentDB automatisch die VPC-Sicherheitsgruppe für Ihre EC2-Instance und für Ihre Amazon DocumentDB-Datenbank.

Die folgenden Anforderungen gelten für die Verbindung einer EC2-Instance mit einer Amazon DocumentDB-Datenbank:

- Die EC2-Instance muss in derselben VPC wie die Amazon DocumentDB-Datenbank vorhanden sein.

Wenn keine EC2-Instances in derselben VPC vorhanden sind, dann bietet die Konsole einen Link zum Erstellen einer solchen Instance.

- Der Benutzer, der die Verbindung einrichtet, muss über Berechtigungen zum Ausführen der folgenden Amazon-EC2-Vorgänge verfügen:
  - `ec2:AuthorizeSecurityGroupEgress`
  - `ec2:AuthorizeSecurityGroupIngress`
  - `ec2:CreateSecurityGroup`
  - `ec2:DescribeInstances`
  - `ec2:DescribeNetworkInterfaces`
  - `ec2:DescribeSecurityGroups`
  - `ec2:ModifyNetworkInterfaceAttribute`
  - `ec2:RevokeSecurityGroupEgress`

Auf Ihrem Konto fallen ggf. Kosten über Availability Zones hinweg an, wenn sich die DB-Instance und die EC2-Instance in unterschiedlichen Availability Zones befinden.

Wenn Sie eine Verbindung zu einer EC2-Instance einrichten, handelt Amazon DocumentDB gemäß der aktuellen Konfiguration der Sicherheitsgruppen, die der Amazon DocumentDB-Datenbank und der EC2-Instance zugeordnet sind, wie in der folgenden Tabelle beschrieben:

Aktuelle Konfiguration der Amazon DocumentDB-Sicherheitsgruppe	Aktuelle EC2-Sicherheitsgruppenkonfiguration	Amazon DocumentDB-Aktion
Der Amazon DocumentDB-Datenbank sind eine oder mehrere Sicherheitsgruppen zugeordnet, deren Name dem Muster entspricht. `DocumentDB-ec2-n` Eine Sicherheitsgruppe, die dem Muster entspricht, wurde nicht geändert. Diese Sicherheitsgruppe enthält nur	Der EC2-Instance sind eine oder mehrere Sicherheitsgruppen zugeordnet, deren Name dem Muster entspricht `DocumentDB-ec2-n` (wobei n eine Zahl ist). Eine Sicherheitsgruppe, die dem Muster entspricht, wurde nicht geändert. Diese Sicherheitsgruppe hat nur eine Regel	Amazon DocumentDB ergreift keine Maßnahmen. Eine Verbindung zwischen der EC2-Instance und der Amazon DocumentDB-Datenbank wurde bereits automatisch konfiguriert. Da bereits eine Verbindung zwischen der EC2-Instance und der Amazon DocumentDB-Datenbank

Aktuelle Konfiguration der Amazon DocumentDB-Sicherheitsgruppe	Aktuelle EC2-Sicherheitsgruppenkonfiguration	Amazon DocumentDB-Aktion
eine Regel für eingehenden Datenverkehr mit der VPC-Sicherheitsgruppe der EC2-Instance als Quelle.	für ausgehenden Datenverkehr mit der VPC-Sicherheitsgruppe der Amazon DocumentDB-Datenbank als Quelle.	besteht, werden die Sicherheitsgruppen nicht geändert.

Aktuelle Konfiguration der Amazon DocumentDB-Sicherheitsgruppe	Aktuelle EC2-Sicherheitsgruppenkonfiguration	Amazon DocumentDB-Aktion
Es gilt eine der folgenden Bedingungen: • Der Amazon DocumentDB-Datenbank ist keine Sicherheitsgruppe mit einem Namen zugeordnet, der dem Muster entspricht. `DocumentDB-ec2-n` • Der Amazon DocumentDB sind eine oder mehrere Sicherheitsgruppen zugeordnet, deren Name dem Muster entspricht. `DocumentDB-ec2-n` Amazon DocumentDB kann jedoch keine dieser Sicherheitsgruppen für die Verbindung mit der EC2-Instance verwenden. Amazon DocumentDB kann keine Sicherheitsgruppe verwenden, die keine Regel für eingehenden Datenverkehr mit der VPC-Sicherheitsgruppe der EC2-Instance als Quelle hat. Amazon DocumentDB kann auch keine Sicherheitsgruppe verwenden, die geändert wurde. Beispiele für Änderungen sind das Hinzufügen einer Regel	Es gilt eine der folgenden Bedingungen: • Der EC2-Instance ist keine Sicherheitsgruppe zugeordnet, deren Name dem Muster `ec2-DocumentDB-n` entspricht. • Der EC2-Instance sind eine oder mehrere Sicherheitsgruppen zugeordnet, deren Name dem Muster `ec2-DocumentDB-n` entspricht. Amazon DocumentDB kann jedoch keine dieser Sicherheitsgruppen für die Verbindung mit der Amazon DocumentDB-Datenbank verwenden. Amazon DocumentDB kann keine Sicherheitsgruppe verwenden, die keine Regel für ausgehenden Datenverkehr mit der VPC-Sicherheitsgruppe der Amazon DocumentDB-Datenbank als Quelle hat. Amazon DocumentDB kann auch keine Sicherheitsgruppe verwenden, die geändert wurde.	Amazon DocumentDB-Aktion: Erstellen Sie neue Sicherheitsgruppen

Aktuelle Konfiguration der Amazon DocumentDB-Sicherheitsgruppe	Aktuelle EC2-Sicherheitsgruppenkonfiguration	Amazon DocumentDB-Aktion
oder das Ändern des Ports einer vorhandenen Regel.		
Der Amazon DocumentDB-Datenbank sind eine oder mehrere Sicherheitsgruppen zugeordnet, deren Name dem Muster entspricht. `DocumentDB-ec2-n` Eine Sicherheitsgruppe, die dem Muster entspricht, wurde nicht geändert. Diese Sicherheitsgruppe enthält nur eine Regel für eingehenden Datenverkehr mit der VPC-Sicherheitsgruppe der EC2-Instance als Quelle.	Der EC2-Instance sind eine oder mehrere Sicherheitsgruppen zugeordnet, deren Name dem Muster `ec2-DocumentDB-n` entspricht. Amazon DocumentDB kann jedoch keine dieser Sicherheitsgruppen für die Verbindung mit der Amazon DocumentDB-Datenbank verwenden. Amazon DocumentDB kann keine Sicherheitsgruppe verwenden, die keine Regel für ausgehenden Datenverkehr mit der VPC-Sicherheitsgruppe der Amazon DocumentDB-Datenbank als Quelle hat. Amazon DocumentDB kann auch keine Sicherheitsgruppe verwenden, die geändert wurde.	Amazon DocumentDB-Aktion: Erstellen Sie neue Sicherheitsgruppen

Aktuelle Konfiguration der Amazon DocumentDB-Sicherheitsgruppe	Aktuelle EC2-Sicherheitsgruppenkonfiguration	Amazon DocumentDB-Aktion
Der Amazon DocumentDB-Datenbank sind eine oder mehrere Sicherheitsgruppen zugeordnet, deren Name dem Muster entspricht. `DocumentDB-ec2-n` Eine Sicherheitsgruppe, die dem Muster entspricht, wurde nicht geändert. Diese Sicherheitsgruppe enthält nur eine Regel für eingehenden Datenverkehr mit der VPC-Sicherheitsgruppe der EC2-Instance als Quelle.	Eine gültige EC2-Sicherheitsgruppe für die Verbindung ist vorhanden, jedoch nicht mit der EC2-Instance verknüpft. Die Sicherheitsgruppe trägt einen Namen, der dem Muster `DocumentDB-ec2-n` entspricht. Sie wurde nicht geändert. Es gibt nur eine Regel für ausgehenden Datenverkehr mit der VPC-Sicherheitsgruppe der Amazon DocumentDB-Datenbank als Quelle.	Amazon DocumentDB-Aktion: Ordnen Sie die EC2-Sicherheitsgruppe zu

Aktuelle Konfiguration der Amazon DocumentDB-Sicherheitsgruppe	Aktuelle EC2-Sicherheitsgruppenkonfiguration	Amazon DocumentDB-Aktion
Es gilt eine der folgenden Bedingungen: • Der Amazon DocumentDB-Datenbank ist keine Sicherheitsgruppe mit einem Namen zugeordnet, der dem Muster entspricht. `DocumentDB-ec2-n` • Der Amazon DocumentDB-Datenbank sind eine oder mehrere Sicherheitsgruppen zugeordnet, deren Name dem Muster entspricht. `DocumentDB-ec2-n` Amazon DocumentDB kann jedoch keine dieser Sicherheitsgruppen für die Verbindung mit der EC2-Instance verwenden. Amazon DocumentDB kann keine Sicherheitsgruppe verwenden, die keine Regel für eingehenden Datenverkehr mit der VPC-Sicherheitsgruppe der EC2-Instance als Quelle hat. Amazon DocumentDB kann auch keine Sicherheitsgruppe verwenden, die geändert wurde.	Der EC2-Instance sind eine oder mehrere Sicherheitsgruppen zugeordnet, deren Name dem Muster `DocumentDB-ec2-n` entspricht. Eine Sicherheitsgruppe, die dem Muster entspricht, wurde nicht geändert. Diese Sicherheitsgruppe hat nur eine Regel für ausgehenden Datenverkehr mit der VPC-Sicherheitsgruppe der Amazon DocumentDB-Datenbank als Quelle.	Amazon DocumentDB-Aktion: Erstellen Sie neue Sicherheitsgruppen

Amazon DocumentDB-Aktion: Neue Sicherheitsgruppen erstellen

Amazon DocumentDB ergreift die folgenden Aktionen:

- Erstellt eine neue Sicherheitsgruppe, die dem Muster `DocumentDB-ec2-n` entspricht. Diese Sicherheitsgruppe enthält eine Regel für eingehenden Datenverkehr mit der VPC-Sicherheitsgruppe der EC2-Instance als Quelle. Diese Sicherheitsgruppe ist der Amazon DocumentDB-Datenbank zugeordnet und ermöglicht der EC2-Instance den Zugriff auf die Amazon DocumentDB-Datenbank.
- Erstellt eine neue Sicherheitsgruppe, die dem Muster `ec2-DocumentDB-n` entspricht. Diese Sicherheitsgruppe hat eine Regel für ausgehenden Datenverkehr mit der VPC-Sicherheitsgruppe der Amazon DocumentDB-Datenbank als Quelle. Diese Sicherheitsgruppe ist der EC2-Instance zugeordnet und ermöglicht es der EC2-Instance, Datenverkehr an die Amazon DocumentDB-Datenbank zu senden.

Amazon DocumentDB-Aktion: Ordnen Sie der EC2-Sicherheitsgruppe zu

Amazon DocumentDB ordnet der EC2-Instance die gültige, vorhandene EC2-Sicherheitsgruppe zu. Diese Sicherheitsgruppe ermöglicht es der EC2-Instance, Datenverkehr an die Amazon DocumentDB-Datenbank zu senden.

Anzeigen verbundener Rechenressourcen

Sie können das verwenden AWS-Managementkonsole , um die Rechenressourcen anzuzeigen, die mit einer Amazon DocumentDB-Datenbank verbunden sind. Zu den angezeigten Ressourcen gehören Rechenressourcenverbindungen, die automatisch eingerichtet wurden. Sie können die Konnektivität mit Rechenressourcen auf folgende Weise automatisch einrichten:

- Sie können die Rechenressource auswählen, wenn Sie die Datenbank erstellen. Weitere Informationen finden Sie unter Erstellen eines Amazon DocumentDB-Clusters und Erstellen eines Multi-AZ DB-Clusters.
- Sie können die Konnektivität zwischen einer vorhandenen Datenbank und einer Rechenressource einrichten. Weitere Informationen finden Sie unter Stellen Sie automatisch eine Verbindung zu Amazon EC2 her.

Die aufgelisteten Rechenressourcen enthalten keine Ressourcen, die manuell mit der Datenbank verbunden wurden. Sie können beispielsweise einer Rechenressource den manuellen Zugriff auf eine

Datenbank erlauben, indem Sie der VPC-Sicherheitsgruppe, die der Datenbank zugeordnet ist, eine Regel hinzufügen.

Für die Auflistung einer Rechenressource müssen die folgenden Bedingungen erfüllt sei:

- Der Name der Sicherheitsgruppe, die der Rechenressource zugeordnet ist, entspricht dem Muster `ec2-DocumentDB-n` (wobei n eine Zahl ist).
- Die der Rechenressource zugeordnete Sicherheitsgruppe hat eine Regel für ausgehenden Datenverkehr, bei der der Portbereich auf den Port festgelegt ist, den die Amazon DocumentDB-Datenbank verwendet.
- Die der Rechenressource zugeordnete Sicherheitsgruppe hat eine Regel für ausgehenden Datenverkehr, wobei die Quelle auf eine Sicherheitsgruppe gesetzt ist, die der Amazon DocumentDB-Datenbank zugeordnet ist.
- Der Name der Sicherheitsgruppe, die der Amazon DocumentDB-Datenbank zugeordnet ist, entspricht dem Muster `DocumentDB-ec2-n` (wobei n eine Zahl ist).
- Die mit der Amazon DocumentDB-Datenbank verknüpfte Sicherheitsgruppe hat eine Regel für eingehenden Datenverkehr, bei der der Portbereich auf den Port festgelegt ist, den die Amazon DocumentDB-Datenbank verwendet.
- Die mit der Amazon DocumentDB-Datenbank verknüpfte Sicherheitsgruppe hat eine Regel für eingehenden Datenverkehr, wobei die Quelle auf eine Sicherheitsgruppe gesetzt ist, die der Rechenressource zugeordnet ist.

Um Rechenressourcen anzuzeigen, die mit einer Amazon DocumentDB-Datenbank verbunden sind

1. Melden Sie sich bei und öffnen Sie die Amazon DocumentDB-Konsole unter. AWS-Managementkonsole https://console.aws.amazon.com/docdb
2. Wählen Sie im Navigationsbereich Datenbanken und dann den Namen der Amazon DocumentDB-Datenbank aus.
3. Sehen Sie sich auf der Registerkarte Konnektivität und Sicherheit die Rechenressourcen im Abschnitt Verbundene Rechenressourcen an.

Stellen Sie eine manuelle Verbindung zu Amazon EC2 her

Topics

- Schritt 1: Erstellen Sie eine Amazon EC2-Instance

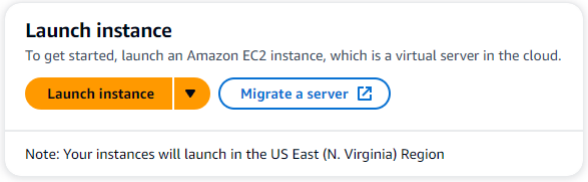
- Schritt 2: Erstellen einer Sicherheitsgruppe
- Schritt 3: Erstellen Sie einen Amazon DocumentDB-Cluster
- Schritt 4: Stellen Sie eine Verbindung zu Ihrer Amazon EC2-Instance her
- Schritt 5: Installieren Sie die MongoDB Shell
- Schritt 6: Amazon DocumentDB TLS verwalten
- Schritt 7: Stellen Sie eine Verbindung zu Ihrem Amazon DocumentDB-Cluster her
- Schritt 8: Daten einfügen und abfragen
- Schritt 9: Erkunden

Bei den folgenden Schritten wird davon ausgegangen, dass Sie die Schritte im Voraussetzungen Thema abgeschlossen haben.

Schritt 1: Erstellen Sie eine Amazon EC2-Instance

In diesem Schritt erstellen Sie eine Amazon EC2-Instance in derselben Region und Amazon VPC, die Sie später zur Bereitstellung Ihres Amazon DocumentDB-Clusters verwenden werden.

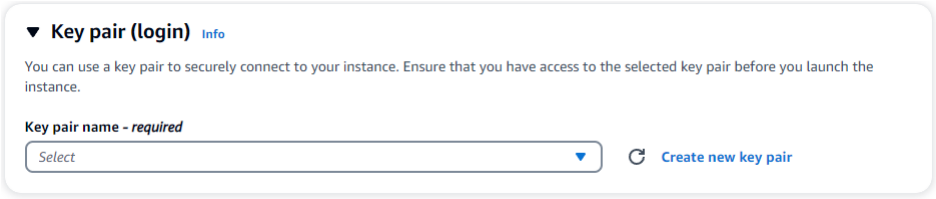
1. Wählen Sie auf der Amazon EC2-Konsole Launch Instance aus.



2. Geben Sie einen Namen oder eine Kennung in das Feld Name ein, das sich im Abschnitt Name und Tags befindet.
3. Suchen Sie in der Dropdownliste Amazon Machine Image (AMI) nach Amazon Linux 2 AMI und wählen Sie es aus.

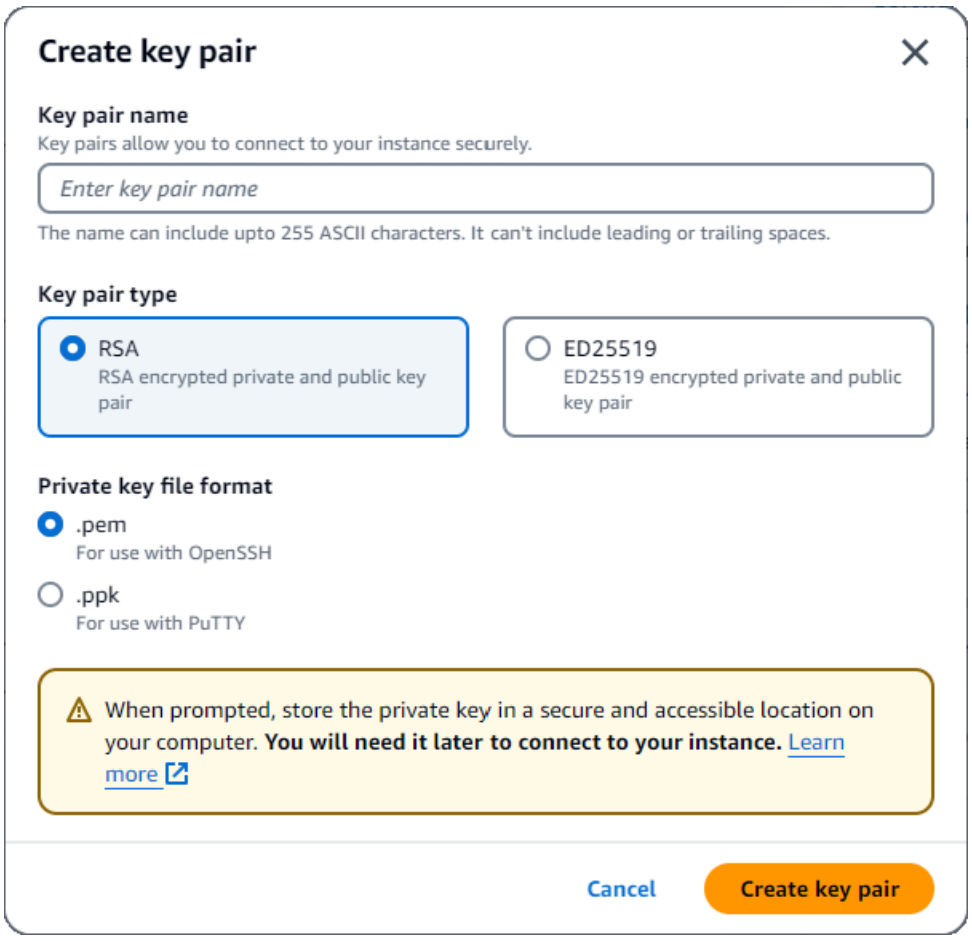


4. Suchen Sie in der Dropdownliste Instance-Typ nach t3.micro und wählen Sie es aus.
5. Geben Sie im Abschnitt Schlüsselpaar (Anmeldung) die ID eines vorhandenen Schlüsselpaars ein, oder wählen Sie Neues Schlüsselpaar erstellen.



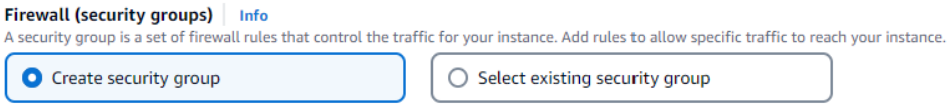
Sie müssen ein Amazon EC2-Schlüsselpaar angeben.

- Wenn Sie ein Amazon EC2-Schlüsselpaar haben:
  1. Wählen Sie ein Schlüsselpaar aus, wählen Sie Ihr Schlüsselpaar aus der Liste aus.
  2. Sie müssen bereits über die private Schlüsseldatei (.pem- oder .ppk-Datei) verfügen, um sich bei Ihrer Amazon EC2-Instance anmelden zu können.
- Wenn Sie kein Amazon EC2-Schlüsselpaar haben:
  1. Wählen Sie Neues Schlüsselpaar erstellen. Das Dialogfeld Schlüsselpaar erstellen wird angezeigt.
  2. Geben Sie einen Namen in das Feld Schlüsselpaar-Name ein.
  3. Wählen Sie den Schlüsselpaartyp und das Dateiformat für den privaten Schlüssel.
  4. Wählen Sie Create Key Pair (Schlüsselpaar erstellen) aus.



> **Note**
>
> Aus Sicherheitsgründen empfehlen wir dringend, ein Schlüsselpaar sowohl für die SSH- als auch für die Internetverbindung zu Ihrer EC2-Instance zu verwenden.

6. Wählen Sie im Abschnitt Netzwerkeinstellungen unter Firewall (Sicherheitsgruppen) entweder Sicherheitsgruppe erstellen oder Bestehende Sicherheitsgruppe auswählen aus.



Wenn Sie eine vorhandene Sicherheitsgruppe auswählen möchten, wählen Sie eine aus der Dropdownliste Allgemeine Sicherheitsgruppen aus.

Wenn Sie eine neue Sicherheitsgruppe erstellen möchten, gehen Sie wie folgt vor:

a. Überprüfen Sie alle Regeln zur Zulassung des Datenverkehrs, die für Ihre EC2-Konnektivität gelten.

b. Wählen Sie im Feld IP Meine IP oder Benutzerdefiniert aus, um aus einer Liste von CIDR-Blöcken, Präfixlisten oder Sicherheitsgruppen auszuwählen. Wir empfehlen Anywhere nicht als Option, es sei denn, Ihre EC2-Instance befindet sich in einem isolierten Netzwerk, da es allen IP-Adressen den Zugriff auf Ihre EC2-Instance ermöglicht.

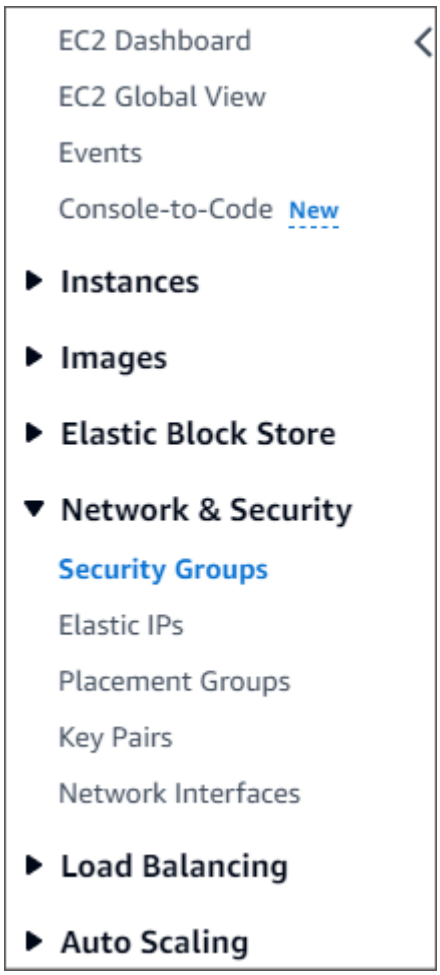


7. Überprüfen Sie im Abschnitt Zusammenfassung Ihre EC2-Konfiguration und wählen Sie Launch Instance, falls dies korrekt ist.

Schritt 2: Erstellen einer Sicherheitsgruppe

Sie erstellen jetzt eine neue Sicherheitsgruppe in Ihrer standardmäßigen Amazon-VPC. Die Sicherheitsgruppe `demoDocDB` ermöglicht es Ihnen, von Ihrer Amazon EC2-Instance aus auf Port 27017 (dem Standardport für Amazon DocumentDB) eine Verbindung zu Ihrem Amazon DocumentDB-Cluster herzustellen.

1. Wählen Sie in der Amazon EC2-Managementkonsole unter Netzwerk und Sicherheit die Option Sicherheitsgruppen aus.



2. Wählen Sie Sicherheitsgruppe erstellen aus.



3. Im Abschnitt Grundlegende Details:

   a. Geben Sie für Security group name (Name der Sicherheitsgruppe) `demoDocDB` ein.

   b. Geben Sie im Feld Description (Beschreibung) eine Beschreibung ein.

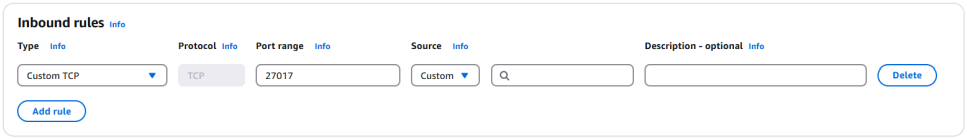
   c. Akzeptieren Sie für VPC die Verwendung Ihrer Standard-VPC.

4. Wählen Sie im Abschnitt Eingehende Regeln die Option Regel hinzufügen aus.

   a. Wählen Sie für Typ die Option Benutzerdefinierte TCP-Regel (Standard) aus.

   b. Geben Sie für Portbereich den Wert ein`27017`.

   c. Wählen Sie für Source (Quelle) die Option Custom (Benutzerdefiniert) aus. Suchen Sie im Feld daneben nach der Sicherheitsgruppe, die Sie gerade in Schritt 1 erstellt haben. Möglicherweise müssen Sie Ihren Browser aktualisieren, damit die Amazon EC2-Konsole den Quellnamen automatisch ausfüllt.



5. Übernehmen Sie alle anderen Standardeinstellungen und wählen Sie Sicherheitsgruppe erstellen.



Schritt 3: Erstellen Sie einen Amazon DocumentDB-Cluster

Während die Amazon EC2-Instance bereitgestellt wird, erstellen Sie Ihren Amazon DocumentDB-Cluster.

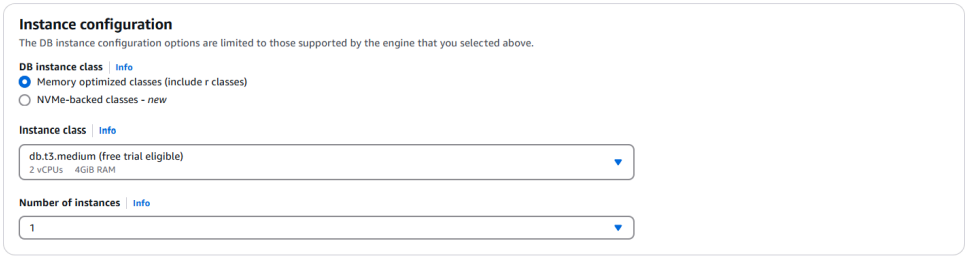
1. Navigieren Sie zur Amazon DocumentDB-Konsole und wählen Sie im Navigationsbereich Clusters aus.
2. Wählen Sie Erstellen aus.
3. Belassen Sie die Einstellung für den Clustertyp auf der Standardeinstellung Instance Based Cluster.
4. Geben Sie in der Cluster-Konfiguration als Cluster-ID einen eindeutigen Namen ein. Beachten Sie, dass die Konsole alle Clusternamen in Kleinbuchstaben umwandelt, unabhängig davon, wie sie eingegeben werden.

   Belassen Sie die Engine-Version auf dem Standardwert 5.0.0.
5. Belassen Sie für die Cluster-Speicherkonfiguration die Standardeinstellung von Amazon DocumentDB Standard.
6. In der Instanzkonfiguration:

   - Wählen Sie für die DB-Instance-Klasse die Option Speicheroptimierte Klassen (einschließlich R-Klassen) aus (dies ist die Standardeinstellung).

     Die andere Instanzoption sind NVMe-backed Klassen. Weitere Informationen hierzu finden Sie unter NVMe-backed Instanzen.
   - Wählen Sie für die Instance-Klasse den Instanztyp aus, der Ihren Anforderungen entspricht. Eine detailliertere Erläuterung der Instanzklassen finden Sie unterSpezifikationen der Instanzklasse.

- Wählen Sie für die Anzahl der Instances eine Zahl, die Ihren Anforderungen am besten entspricht. Denken Sie daran, je niedriger die Anzahl, desto niedriger sind die Kosten und desto geringer ist das read/write Volumen, das vom Cluster verwaltet werden kann.



7. Belassen Sie für Konnektivität die Standardeinstellung Keine Verbindung zu einer EC2-Rechenressource herstellen.

> **Note**
>
> Wenn Sie eine Verbindung zu einer EC2-Rechenressource herstellen, werden automatisch Sicherheitsgruppen für Ihre Verbindung zu Ihrem Cluster erstellt. Da Sie diese Sicherheitsgruppen im vorherigen Schritt manuell erstellt haben, sollten Sie Keine Verbindung zu einer EC2-Rechenressource herstellen auswählen, um keinen zweiten Satz von Sicherheitsgruppen zu erstellen.

8. Geben Sie im Abschnitt Authentifizierung einen Benutzernamen für den Hauptbenutzer ein und wählen Sie dann Selbstverwaltet aus. Geben Sie ein Passwort ein und bestätigen Sie es.

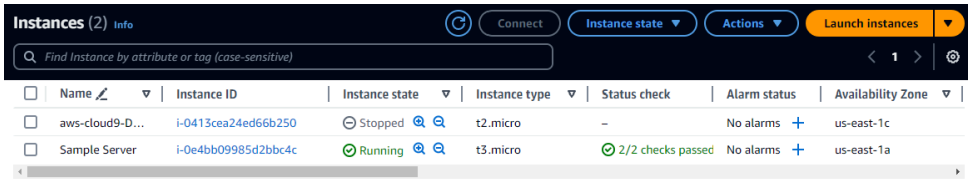
   Wenn Sie stattdessen Verwaltet in ausgewählt haben AWS Secrets Manager, finden Sie Passwortverwaltung mit Amazon DocumentDB und AWS Secrets Manager weitere Informationen unter.

9. Wählen Sie Cluster erstellen.

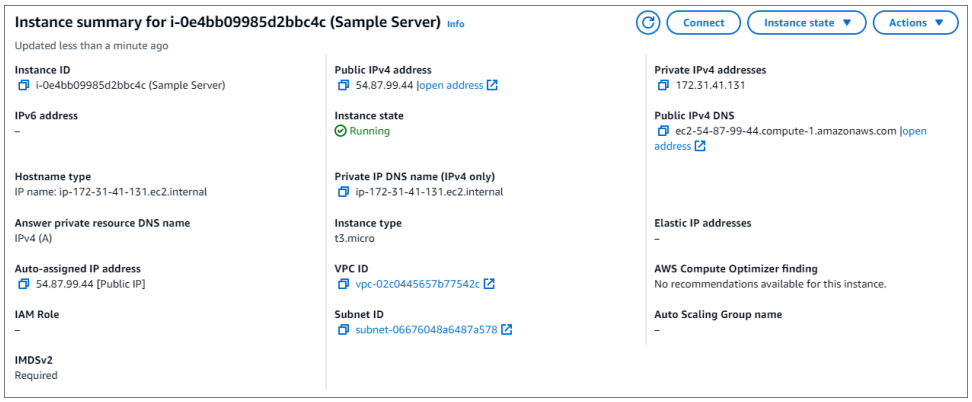
Schritt 4: Stellen Sie eine Verbindung zu Ihrer Amazon EC2-Instance her

Wenn Sie eine Verbindung zu Ihrer Amazon EC2-Instance herstellen, können Sie die MongoDB-Shell installieren. Durch die Installation der Mongo-Shell können Sie eine Verbindung zu Ihrem Amazon DocumentDB-Cluster herstellen und diesen abfragen. Führen Sie folgende Schritte aus:

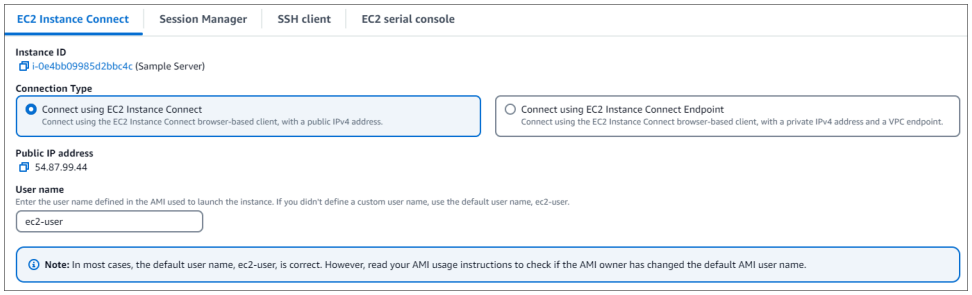
1. Navigieren Sie auf der Amazon EC2-Konsole zu Ihren Instances und prüfen Sie, ob die Instance, die Sie gerade erstellt haben, läuft. Wenn ja, wählen Sie die Instance aus, indem Sie auf die Instance-ID klicken.



2. Wählen Sie Connect aus.



3. Es gibt vier Optionen mit Registerkarten für Ihre Verbindungsmethode: Amazon EC2 Instance Connect, Session Manager, SSH-Client oder serielle EC2-Konsole. Sie müssen eine auswählen und den Anweisungen folgen. Wenn Sie fertig sind, wählen Sie Verbinden.



> **Note**
>
> Wenn sich Ihre IP-Adresse geändert hat, nachdem Sie mit diesem Walkthrough begonnen haben, oder Sie zu einem späteren Zeitpunkt zu Ihrer Umgebung zurückkehren, müssen Sie Ihre demoEC2 Sicherheitsgruppenregel für eingehenden Datenverkehr aktualisieren, um eingehenden Datenverkehr von Ihrer neuen API-Adresse zu ermöglichen.

Schritt 5: Installieren Sie die MongoDB Shell

Sie können jetzt die MongoDB-Shell installieren, ein Befehlszeilenprogramm, mit dem Sie Ihren Amazon DocumentDB-Cluster verbinden und abfragen. Derzeit gibt es zwei Versionen der MongoDB-Shell: die neueste Version, Mongosh, und die vorherige Version, Mongo-Shell.

> **Important**
>
> Es gibt eine bekannte Einschränkung bei Node.js Treibern, die älter als Version 6.13.1 sind und die derzeit von der IAM-Identitätsauthentifizierung für Amazon DocumentDB nicht unterstützt werden. Node.js Treiber und Tools, die Node.js Treiber verwenden (z. B. Mongosh), müssen aktualisiert werden, um die Node.js Treiberversion 6.13.1 oder höher zu verwenden.

Folgen Sie den nachstehenden Anweisungen, um die MongoDB-Shell für Ihr Betriebssystem zu installieren.

On Amazon Linux

Um die MongoDB-Shell auf Amazon Linux zu installieren

Wenn Sie IAM nicht verwenden und die neueste MongoDB-Shell (mongosh) verwenden möchten, um eine Verbindung zu Ihrem Amazon DocumentDB-Cluster herzustellen, gehen Sie wie folgt vor:

1. Erstellen Sie die Repository-Datei. Führen Sie in der Befehlszeile Ihrer EC2-Instance, die Sie erstellt haben, den folgenden Befehl aus:

```
echo -e "[mongodb-org-5.0] \nname=MongoDB Repository\nbaseurl=https://
repo.mongodb.org/yum/amazon/2023/mongodb-org/5.0/x86_64/\ngpgcheck=1 \nenabled=1
 \ngpgkey=https://pgp.mongodb.com/server-5.0.asc" | sudo tee /etc/yum.repos.d/
mongodb-org-5.0.repo
```

2. Wenn der Vorgang abgeschlossen ist, installieren Sie mongosh mit einer der beiden folgenden Befehlsoptionen an der Befehlszeile:

   Option 1 — Wenn Sie während der Amazon EC2-Konfiguration das standardmäßige Amazon Linux 2023 ausgewählt haben, geben Sie diesen Befehl ein:

```
sudo yum install -y mongodb-mongosh-shared-openssl3
```

Option 2 — Wenn Sie während der Amazon EC2-Konfiguration Amazon Linux 2 ausgewählt haben, geben Sie diesen Befehl ein:

```
sudo yum install -y mongodb-mongosh
```

Wenn Sie IAM verwenden, müssen Sie die vorherige Version der MongoDB-Shell (5.0) verwenden, um eine Verbindung zu Ihrem Amazon DocumentDB-Cluster herzustellen. Gehen Sie wie folgt vor:

1. Erstellen Sie die Repository-Datei. Führen Sie in der Befehlszeile Ihrer EC2-Instance, die Sie erstellt haben, den folgenden Befehl aus:

```
echo -e "[mongodb-org-5.0] \nname=MongoDB Repository\nbaseurl=https://
repo.mongodb.org/yum/amazon/2023/mongodb-org/5.0/x86_64/\ngpgcheck=1 \nenabled=1
 \ngpgkey=https://pgp.mongodb.com/server-5.0.asc" | sudo tee /etc/yum.repos.d/
mongodb-org-5.0.repo
```

2. Wenn der Vorgang abgeschlossen ist, installieren Sie die Mongodb 5.0-Shell mit der folgenden Befehlsoption an der Eingabeaufforderung:

```
sudo yum install -y mongodb-org-shell
```

On Ubuntu

Um mongosh auf Ubuntu zu installieren

1. Importieren Sie den öffentlichen Schlüssel, der von dem Paketverwaltungssystem verwendet wird.

```
curl -fsSL https://pgp.mongodb.com/server-5.0.asc | sudo gpg --dearmor -o /usr/
share/keyrings/mongodb-server-5.0.gpg
```

2. Erstellen Sie die Listendatei `mongodb-org-5.0.list` für MongoDB mit dem korrekten Befehl für Ihre Ubuntu-Version.

```
echo "deb [ arch=amd64,arm64 signed-by=/usr/share/keyrings/mongodb-
server-5.0.gpg ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/5.0
 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-5.0.list
```

3. Importieren und aktualisieren Sie die lokale Paketdatenbank mit dem folgenden Befehl:

```
sudo apt-get update
```

4. Installieren Sie Mongosh.

```
sudo apt-get install -y mongodb-mongosh
```

Weitere Informationen zum Installieren von früheren Versionen von MongoDB auf Ihrem Ubuntu-System finden Sie unter Installieren von MongoDB Community Edition auf Ubuntu.

On other operating systems

Informationen zum Installieren der mongo-Shell auf anderen Betriebssystemen finden Sie unter Installieren von MongoDB Community Edition in der MongoDB-Dokumentation.

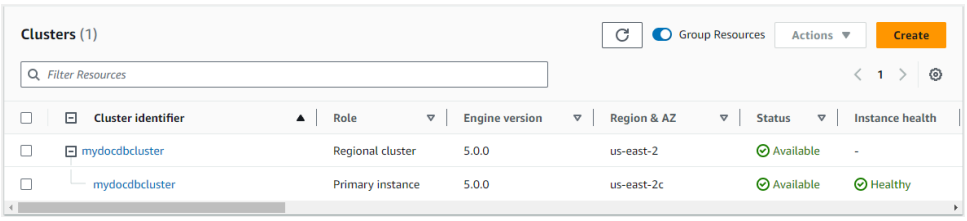
Schritt 6: Amazon DocumentDB TLS verwalten

Laden Sie das CA-Zertifikat für Amazon DocumentDB mit dem folgenden Code herunter: `wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem`

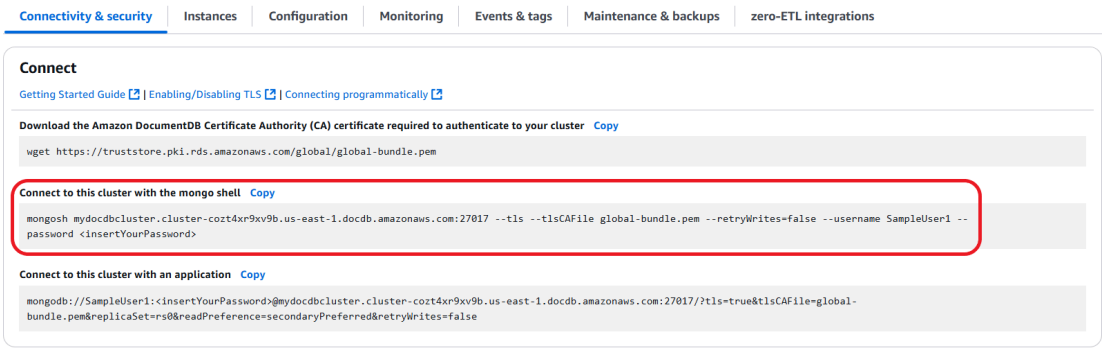
> **Note**
>
> Transport Layer Security (TLS) ist standardmäßig für alle neuen Amazon DocumentDB-Cluster aktiviert. Weitere Informationen finden Sie unter Verschlüsseln von Daten während der Übertragung.

Schritt 7: Stellen Sie eine Verbindung zu Ihrem Amazon DocumentDB-Cluster her

1. Suchen Sie auf der Amazon DocumentDB-Konsole unter Clusters nach Ihrem Cluster. Wählen Sie den Cluster aus, den Sie erstellt haben, indem Sie auf die Cluster-ID für diesen Cluster klicken.



2. Suchen Sie auf der Registerkarte „Konnektivität und Sicherheit“ im Feld „Verbinden“ nach „Mit diesem Cluster verbinden“ mit der Mongo-Shell:



   Kopieren Sie die bereitgestellte Verbindungszeichenfolge und fügen Sie sie in Ihr Terminal ein.

   Nehmen Sie die folgenden Änderungen daran vor:

   a. Stellen Sie sicher, dass Sie den richtigen Benutzernamen in der Zeichenfolge haben.

   b. Lassen Sie es weg, `<insertYourPassword>` damit Sie bei der Verbindung von der Mongo-Shell nach dem Passwort gefragt werden.

   c. Optional: Wenn Sie die IAM-Authentifizierung verwenden oder die vorherige Version der MongoDB-Shell verwenden, ändern Sie Ihre Verbindungszeichenfolge wie folgt:

      ```
      mongo --ssl --host mydocdbcluster.cluster-cozt4xr9xv9b.us-east-1.docdb.amazonaws.com:27017 --sslCAFile global-bundle.pem --username SampleUser1 --password
      ```

      `mydocdbcluster.cluster-cozt4xr9xv9b.us-east-1`Ersetzen Sie es durch dieselben Informationen aus Ihrem Cluster.

3. Drücken Sie in Ihrem Terminal die Eingabetaste. Sie werden jetzt nach Ihrem Passwort gefragt. Geben Sie Ihr Passwort ein.

4. Wenn Sie Ihr Passwort eingeben und die `rs0 [direct: primary] <env-name>>` Aufforderung sehen, sind Sie erfolgreich mit Ihrem Amazon DocumentDB-Cluster verbunden.

Informationen zur Lösung von Verbindungsproblemen finden Sie unterProblembehandlung bei Amazon DocumentDB.

Schritt 8: Daten einfügen und abfragen

Jetzt, da Sie mit Ihrem Cluster verbunden sind, können Sie einige Abfragen ausführen, um sich mit der Verwendung einer Dokumentdatenbank vertraut zu machen.

1. Um ein einzelnes Dokument einzufügen, geben Sie Folgendes ein:

```
db.collection.insertOne({"hello":"DocumentDB"})
```

   Sie erhalten die folgende Ausgabe:

```
{
  acknowledged: true,
  insertedId: ObjectId('673657216bdf6258466b128c')
}
```

2. Sie können das Dokument lesen, das Sie mit dem `findOne()` Befehl geschrieben haben (da nur ein einziges Dokument zurückgegeben wird). Geben Sie Folgendes ein:

```
db.collection.findOne()
```

   Sie erhalten die folgende Ausgabe:

```
{ "_id" : ObjectId("5e401fe56056fda7321fbd67"), "hello" : "DocumentDB" }
```

3. Um ein paar weitere Abfragen durchzuführen, ziehen Sie einen Anwendungsfall mit Spielprofilen in Betracht. Fügen Sie zunächst einige Einträge in eine Sammlung mit dem Titel ein`profiles`. Geben Sie Folgendes ein:

```
db.profiles.insertMany([{ _id: 1, name: 'Matt', status: 'active', level: 12, score:
 202 },
      { _id: 2, name: 'Frank', status: 'inactive', level: 2, score: 9 },
      { _id: 3, name: 'Karen', status: 'active', level: 7, score: 87 },
      { _id: 4, name: 'Katie', status: 'active', level: 3, score: 27 }
])
```

   Sie erhalten die folgende Ausgabe:

```
{ acknowledged: true, insertedIds: { '0': 1, '1': 2, '2': 3, '3': 4 } }
```

4. Verwenden Sie den `find()` Befehl, um alle Dokumente in der Profilsammlung zurückzugeben. Geben Sie Folgendes ein:

   ```
   db.profiles.find()
   ```

   Sie erhalten eine Ausgabe, die den Daten entspricht, die Sie in Schritt 3 eingegeben haben.

5. Verwenden Sie mithilfe eines Filters eine Abfrage für ein einzelnes Dokument. Geben Sie Folgendes ein:

   ```
   db.profiles.find({name: "Katie"})
   ```

   Sie erhalten die folgende Ausgabe:

   ```
   { "_id" : 4, "name" : "Katie", "status": "active", "level": 3, "score":27}
   ```

6. Versuchen wir nun, ein Profil zu finden und es mit dem `findAndModify` Befehl zu ändern. Wir geben dem Benutzer Matt zusätzliche 10 Punkte mit dem folgenden Code:

   ```
   db.profiles.findAndModify({
           query: { name: "Matt", status: "active"},
           update: { $inc: { score: 10 } }
       })
   ```

   Du erhältst die folgende Ausgabe (beachte, dass seine Punktzahl noch nicht gestiegen ist):

   ```
   {
       [{_id : 1, name : 'Matt', status: 'active', level: 12, score: 202}]
   ```

7. Mit der folgenden Abfrage können Sie überprüfen, ob sich sein Ergebnis geändert hat:

   ```
   db.profiles.find({name: "Matt"})
   ```

   Sie erhalten die folgende Ausgabe:

   ```
   { "_id" : 1, "name" : "Matt", "status" : "active", "level" : 12, "score" : 212 }
   ```

Schritt 9: Erkunden

Herzlichen Glückwunsch! Sie haben die Schnellstartanleitung für Amazon DocumentDB erfolgreich abgeschlossen.

Was kommt als Nächstes? Erfahren Sie, wie Sie diese leistungsstarke Datenbank mit einigen ihrer beliebten Funktionen voll ausschöpfen können:

- Verwaltung von Amazon DocumentDB-Clustern und -Instances
- Skalierung
- Sichern und Wiederherstellen

> **Note**
>
> Um Kosten zu sparen, können Sie entweder Ihren Amazon DocumentDB-Cluster stoppen, um die Kosten zu senken, oder den Cluster löschen. Standardmäßig stoppt Ihre AWS Cloud9 Umgebung nach 30 Minuten Inaktivität die zugrunde liegende Amazon EC2-Instance.

Stellen Sie mithilfe des Amazon DocumentDB-JDBC-Treibers eine Verbindung her

Der JDBC-Treiber für Amazon DocumentDB bietet eine SQL-relational Schnittstelle für Entwickler und ermöglicht die Konnektivität von BI-Tools wie Tableau und. DbVisualizer

Weitere Informationen finden Sie in der Amazon DocumentDB-JDBC-Treiberdokumentation unter. GitHub

Themen

- Erste Schritte
- Stellen Sie von Tableau Desktop aus eine Verbindung zu Amazon DocumentDB her
- Stellen Sie eine Verbindung zu Amazon DocumentDB her von DbVisualizer
- Automatische JDBC-Schemagenerierung
- SQL-Unterstützung und Einschränkungen
- Fehlerbehebung

Erste Schritte

Schritt 1. Amazon DocumentDB-Cluster erstellen

Wenn Sie noch keinen Amazon DocumentDB-Cluster erstellt haben, erstellen Sie einen mithilfe der Anweisungen in der. Erste Schritte mit Amazon DocumentDB

> **Note**
>
> Amazon DocumentDB ist ein reiner Virtual Private Cloud (VPC) -Service. Wenn Sie eine Verbindung von einem lokalen Computer außerhalb der VPC des Clusters herstellen, müssen Sie eine SSH-Verbindung zu einer Amazon EC2-Instance herstellen. Starten Sie in diesem Fall Ihren Cluster mithilfe der Anweisungen unter. Stellen Sie mithilfe von Amazon EC2 eine Verbindung her Weitere Informationen zum SSH-Tunneling und wann Sie es möglicherweise benötigen, finden Sie unter Verwenden eines SSH-Tunnels zum Herstellen einer Verbindung zu Amazon DocumentDB.

Schritt 2. JRE- oder JDK-Installation

Abhängig von Ihrer BI-Anwendung müssen Sie möglicherweise sicherstellen, dass eine 64-Bit-JRE- oder JDK-Installation der Version 8 oder höher auf Ihrem Computer installiert ist. Sie können das Java SE Runtime Environment 8 hier herunterladen.

Schritt 3. Laden Sie den DocumentDB-JDBC-Treiber herunter

Laden Sie den DocumentDB-JDBC-Treiber von hier herunter. https://github.com/aws/amazon-documentdb-jdbc-driver/releases Der Treiber ist als einzelne JAR-Datei verpackt (z. B. documentdb-jdbc-1.0.0-all.jar).

Schritt 4. Verwenden Sie einen SSH-Tunnel, um eine Verbindung zu Amazon DocumentDB herzustellen

Amazon DocumentDB-Cluster (mit MongoDB-Kompatibilität) werden in einer Amazon Virtual Private Cloud (Amazon VPC) bereitgestellt. Auf sie kann direkt von Amazon EC2-Instances oder anderen AWS Diensten zugegriffen werden, die in derselben Amazon VPC bereitgestellt werden. Darüber hinaus können EC2a-Instances oder andere AWS Dienste in verschiedenen VPCs in derselben AWS Region oder anderen Regionen über VPC-Peering auf Amazon DocumentDB zugreifen.

Sie können SSH-Tunneling (auch bekannt als Portweiterleitung) verwenden, um von außerhalb der VPC des Clusters auf Ihre Amazon DocumentDB-Ressourcen zuzugreifen. Dies wird bei den meisten Benutzern der Fall sein, die ihre Anwendung nicht auf einer VM in derselben VPC wie der DocumentDB-Cluster ausführen.

Um einen SSH-Tunnel zu erstellen, benötigen Sie eine Amazon EC2-Instance, die in derselben Amazon VPC wie Ihr Amazon DocumentDB-Cluster ausgeführt wird. Sie können entweder eine vorhandene EC2-Instance in derselben VPC wie Ihr Cluster verwenden oder eine erstellen. Sie können einen SSH-Tunnel zum Amazon DocumentDB-Cluster einrichten, `sample-cluster.node.us-east-1.docdb.amazonaws.com` indem Sie den folgenden Befehl auf Ihrem lokalen Computer ausführen.

```
ssh -i "ec2Access.pem" -L 27017:sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017 ubuntu@ec2-34-229-221-164.compute-1.amazonaws.com -
N
```

Das Flag -L wird für die Weiterleitung eines lokalen Ports verwendet. Dies ist eine Voraussetzung für die Verbindung zu einem BI-Tool, das auf einem Client außerhalb Ihrer VPC ausgeführt wird. Sobald Sie den obigen Schritt ausgeführt haben, können Sie mit den nächsten Schritten für das BI-Tool Ihrer Wahl fortfahren.

Weitere Informationen zum SSH-Tunneling finden Sie in der Dokumentation zur Verwendung eines SSH-Tunnels zur Verbindung mit Amazon DocumentDB.

Stellen Sie von Tableau Desktop aus eine Verbindung zu Amazon DocumentDB her

Themen

- Hinzufügen des Amazon DocumentDB-JDBC-Treibers
- Stellen Sie mithilfe von Tableau — SSH-Tunnel eine Verbindung zu Amazon DocumentDB her

Hinzufügen des Amazon DocumentDB-JDBC-Treibers

Um von Tableau Desktop aus eine Verbindung zu Amazon DocumentDB herzustellen, müssen Sie den Amazon DocumentDB-JDBC-Treiber und den DocumentDB Tableau-Connector herunterladen und installieren.

1. Laden Sie die Amazon DocumentDB-JDBC-Treiber-JAR-Datei aus dem Amazon DocumentDB-JDBC-Treiber-Repository herunter und kopieren Sie sie je nach Betriebssystem in eines der folgenden Verzeichnisse:
   - Windows - `C:\Program Files\Tableau\Drivers`
   - macOS - `~/Library/Tableau/Drivers`
2. Laden Sie den DocumentDB Tableau-Connector (eine TACO-Datei) von der Tableau Exchange-Website herunter und kopieren Sie ihn in Ihr My Repository/Connectors Tableau-Verzeichnis.
   - Windows - `C:\Users\[user]\Documents\My Tableau Repository\Connectors`
   - macOS - `/Users/[user]/Documents/My Tableau Repository/Connectors`

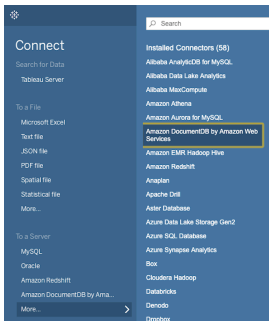
Weitere Informationen finden Sie in der Tableau-Dokumentation.

> **Note**
>
> Wenn Sie neuere CA-Zertifikate verwenden, stellen Sie sicher, dass Sie Ihren JDBC-Treiber auf Version 1.4.5 aktualisieren (in diesem AWS GitHub Repository verfügbar).

Stellen Sie mithilfe von Tableau — SSH-Tunnel eine Verbindung zu Amazon DocumentDB her

Um von einem Client-Computer außerhalb der VPC Ihres DocumentDB-Clusters aus eine Verbindung zu Tableau herzustellen, müssen Sie einen SSH-Tunnel einrichten, bevor Sie die folgenden Schritte ausführen:

1. Starten Sie die Tableau Desktop-Anwendung.
2. Navigieren Sie zu Verbinden > Mit einem Server > Mehr.
3. Wählen Sie unter Installierte Connectors Amazon DocumentDB von Amazon Web Services aus.

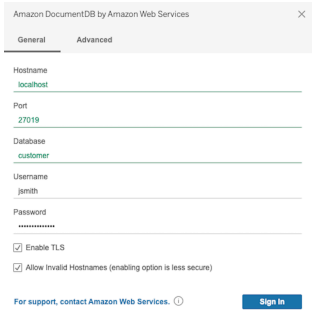


Verbindung zu Amazon DocumentDB mithilfe von Tableau herstellen — externer SSH-Tunnel

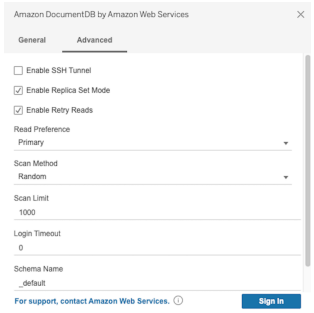
1. Geben Sie die erforderlichen Verbindungsparameter Hostname, Port, Datenbank , Benutzername und Passwort ein. Die Verbindungsparameter im folgenden Beispiel entsprechen der JDBC-Verbindungszeichenfolge:

   ```
   jdbc:documentdb://localhost:27019/test?
   tls=true&tlsAllowInvalidHostnames=true&scanMethod=random&scanLimit=1000&logir
   ```

   die Parameter Benutzername und Passwort separat in einer Eigenschaftensammlung übergeben werden. Weitere Informationen zu Parametern für Verbindungszeichenfolgen finden Sie in der Github-Dokumentation zum Amazon DocumentDB JDBC-Treiber.



2. (Optional) Erweiterte Optionen finden Sie auf der Registerkarte „Erweitert“.

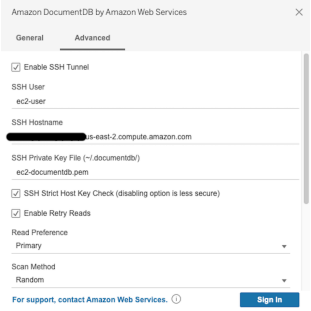


3. Klicken Sie auf Sign in.

Verbindung zu Amazon DocumentDB mithilfe von Tableau herstellen — interner SSH-Tunnel

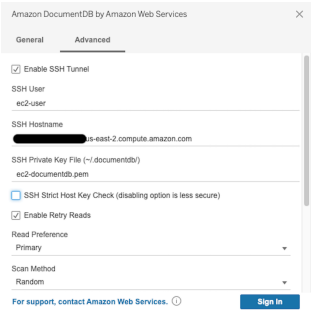
> **Note**
>
> Wenn Sie es vorziehen, den SSH-Tunnel nicht über ein Terminal einzurichten, können Sie die Tableau-GUI verwenden, um Ihre EC2-Instanzdetails anzugeben, die der JDBC-Treiber automatisch zum Erstellen eines SSH-Tunnels verwendet.

1. Wählen Sie auf der Registerkarte Erweitert die Option SSH-Tunnel aktivieren, um weitere Eigenschaften zu überprüfen.



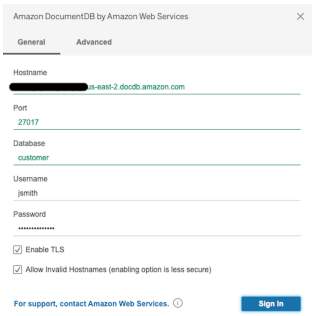
2. Geben Sie den SSH-Benutzer, den SSH-Hostnamen und die private SSH-Schlüsseldatei ein.
3. (Optional) Sie können die SSH-Option Strict Host Key Check deaktivieren, wodurch die Überprüfung des Hostschlüssels anhand einer bekannten Hosts-Datei umgangen wird.

> **Note**
>
> Die Deaktivierung dieser Option ist weniger sicher, da sie zu einem https://en.wikipedia.org/wiki/Man-in-the-middle_attack Man-in-the-Middle-Angriff führen kann.



4. Geben Sie die erforderlichen Parameter ein: Hostname, Port, Datenbank, Benutzername und Passwort.

> **Note**
>
> Stellen Sie sicher, dass Sie den DocumentDB-Cluster-Endpunkt und nicht localhost verwenden, wenn Sie die interne SSH-Tunneloption verwenden.



5. Klicken Sie auf Sign In.

Stellen Sie eine Verbindung zu Amazon DocumentDB her von DbVisualizer

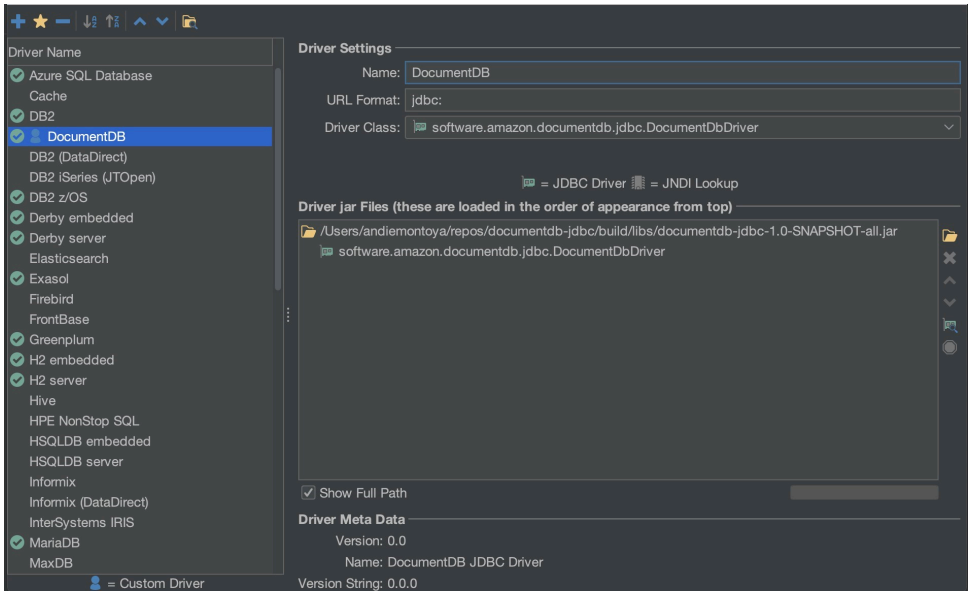
Themen

- Hinzufügen des Amazon DocumentDB-JDBC-Treibers
- Stellen Sie eine Verbindung zu Amazon DocumentDB her mit DbVisualizer

Hinzufügen des Amazon DocumentDB-JDBC-Treibers

Um eine Verbindung zu Amazon DocumentDB herzustellen, müssen DbVisualizer Sie zuerst den Amazon DocumentDB-JDBC-Treiber importieren

1. Starten Sie die DbVisualizer Anwendung und navigieren Sie zum Menüpfad: Tools > Driver Manager...
2. Wählen Sie + (oder wählen Sie im Menü Treiber > Treiber erstellen).
3. Legen Sie Name auf `DocumentDB` fest.
4. Stellen Sie das URL-Format ein auf `jdbc:documentdb://<host>[:port]/<database>[?option=value[&option=value[...]]]`
5. Wählen Sie die Schaltfläche „Ordner“ und dann die JAR-Datei des Amazon DocumentDB-JDBC-Treibers aus und klicken Sie auf die Schaltfläche „Öffnen“.
6. Vergewissern Sie sich, dass das Feld Treiberklasse auf gesetzt ist. `software.amazon.documentdb.jdbc.DocumentDbDriver` Ihre Driver Manager-Einstellungen für DocumentDB sollten wie im folgenden Beispiel aussehen.



7. Schließen Sie das Dialogfeld. Der Amazon DocumentDB-JDBC-Treiber wird eingerichtet und ist einsatzbereit.

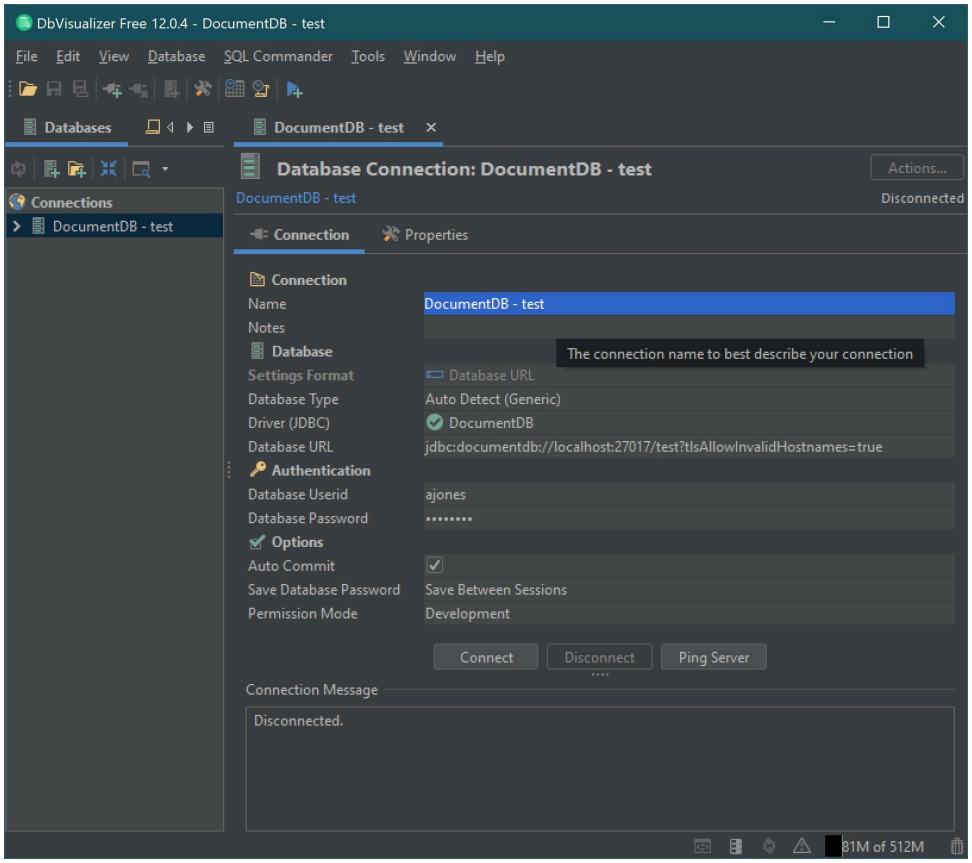
Stellen Sie eine Verbindung zu Amazon DocumentDB her mit DbVisualizer

Stellen Sie eine Verbindung zu Amazon DocumentDB her mit DbVisualizer

1. Wenn Sie eine Verbindung von außerhalb der VPC des Amazon DocumentDB-Clusters herstellen, stellen Sie sicher, dass Sie einen SSH-Tunnel eingerichtet haben.
2. Wählen Sie im Menü der obersten Ebene Datenbank > Datenbankverbindung erstellen.
3. Geben Sie einen beschreibenden Namen für das Feld „Name“ ein.
4. Stellen Sie Driver (JDBC) auf den DocumentDB-Treiber ein, den Sie im vorherigen Abschnitt erstellt haben.
5. Stellen Sie die Datenbank-URL auf Ihre JDBC-Verbindungszeichenfolge ein.

   Beispiel: `jdbc:documentdb://localhost:27017/database?tlsAllowInvalidHostnames=true`
6. Stellen Sie die Datenbank-Benutzer-ID auf Ihre Amazon DocumentDB-Benutzer-ID ein.
7. Setzen Sie das Datenbankkennwort auf das entsprechende Passwort für die Benutzer-ID.

   Ihr Datenbankverbindungsdialogfeld sollte wie das folgende Dialogfeld aussehen:



8. Wählen Sie Connect aus.

Automatische JDBC-Schemagenerierung

Amazon DocumentDB ist eine Dokumentdatenbank und hat daher nicht das Konzept von Tabellen und Schemas. BI-Tools wie Tableau erwarten jedoch, dass die Datenbank, mit der sie verbunden werden, ein Schema präsentiert. Insbesondere wenn die JDBC-Treiberverbindung das Schema für die Sammlung in der Datenbank abrufen muss, werden alle Sammlungen in der Datenbank abgefragt. Der Treiber ermittelt, ob eine zwischengespeicherte Version des Schemas für diese Sammlung bereits existiert. Wenn keine zwischengespeicherte Version vorhanden ist, sucht er in der Sammlung nach Dokumenten und erstellt ein Schema, das auf dem folgenden Verhalten basiert.

Themen

- Einschränkungen bei der Schemagenerierung
- Optionen für die Scanmethode
- Amazon-DocumentDB-Datentypen
- Zuordnung skalarer Dokumentfelder

- Behandlung von Objekt- und Array-Datentypen

Einschränkungen bei der Schemagenerierung

Der DocumentDB-JDBC-Treiber begrenzt die Länge von Bezeichnern auf 128 Zeichen. Der Schemagenator kann die Länge der generierten Bezeichner (Tabellennamen und Spaltennamen) kürzen, um sicherzustellen, dass sie dieser Grenze entsprechen.

Optionen für die Scanmethode

Das Sampling-Verhalten kann mithilfe von Verbindungszeichenfolgen- oder Datenquellenoptionen geändert werden.

- ScanMethode= <option>
  - random — (Standard) — Die Beispieldokumente werden in zufälliger Reihenfolge zurückgegeben.
  - idForward — Die Beispieldokumente werden in der Reihenfolge ihrer ID zurückgegeben.
  - idReverse — Die Beispieldokumente werden in umgekehrter Reihenfolge der ID zurückgegeben.
  - all — Probieren Sie alle Dokumente in der Sammlung aus.
- ScanLimit= <n>— Die Anzahl der Dokumente, für die eine Stichprobe erstellt werden soll. Der Wert muss eine positive ganze Zahl sein. Der Standardwert ist 1000. Wenn ScanMethod auf all gesetzt ist, wird diese Option ignoriert.

Amazon-DocumentDB-Datentypen

Der Amazon DocumentDB-Server unterstützt eine Reihe von MongoDB-Datentypen. Nachfolgend sind die unterstützten Datentypen und die zugehörigen JDBC-Datentypen aufgeführt.

MongoDB-Datentyp	Wird in DocumentDB unterstützt	JDBC-Datentyp
Binäre Daten	Ja	VARBINARY
Boolesch	Ja	BOOLEAN
Double	Ja	DOUBLE

MongoDB-Datentyp	Wird in DocumentDB unterstützt	JDBC-Datentyp
32-Bit-Ganzzahl	Ja	INTEGER
64-Bit-Ganzzahl	Ja	BIGINT
Zeichenfolge	Ja	VARCHAR
ObjectId	Ja	VARCHAR
Date	Ja	TIMESTAMP (ZEITSTEMPEL)
Null	Ja	VARCHAR
Regulärer Ausdruck	Ja	VARCHAR
Zeitstempel	Ja	VARCHAR
MinKey	Ja	VARCHAR
MaxKey	Ja	VARCHAR
Objekt	Ja	virtueller Tisch
Array	Ja	virtueller Tisch
Decimal128	Nein	DECIMAL
JavaScript	Nein	VARCHAR
JavaScript (mit Zielfernrohr)	Nein	VARCHAR
Undefined	Nein	VARCHAR
Symbol	Nein	VARCHAR
dbPointer (4.0+)	Nein	VARCHAR

Zuordnung skalarer Dokumentfelder

Beim Scannen einer Auswahl von Dokumenten aus einer Sammlung erstellt der JDBC-Treiber ein oder mehrere Schemas, um die Beispiele in der Sammlung darzustellen. Im Allgemeinen wird ein Skalarfeld im Dokument einer Spalte im Tabellenschema zugeordnet. In einer Sammlung mit dem Namen team und einem einzelnen Dokument würde dies `{ "_id" : "112233", "name" : "Alastair", "age": 25 }` beispielsweise dem Schema zugeordnet werden:

Tabellenname	Spaltenname	Datentyp	Key (Schlüssel)
Team	Team-ID	VARCHAR	PK
Team	Name	VARCHAR	
Team	age	INTEGER	

Förderung von Konflikten nach Datentyp

Beim Scannen der ausgewählten Dokumente ist es möglich, dass die Datentypen für ein Feld von Dokument zu Dokument nicht konsistent sind. In diesem Fall stuft der JDBC-Treiber den JDBC-Datentyp auf einen gemeinsamen Datentyp um, der für alle Datentypen der Stichprobendokumente geeignet ist.

Zum Beispiel:

```
{
"_id" : "112233",
"name" : "Alastair", "age" : 25
}

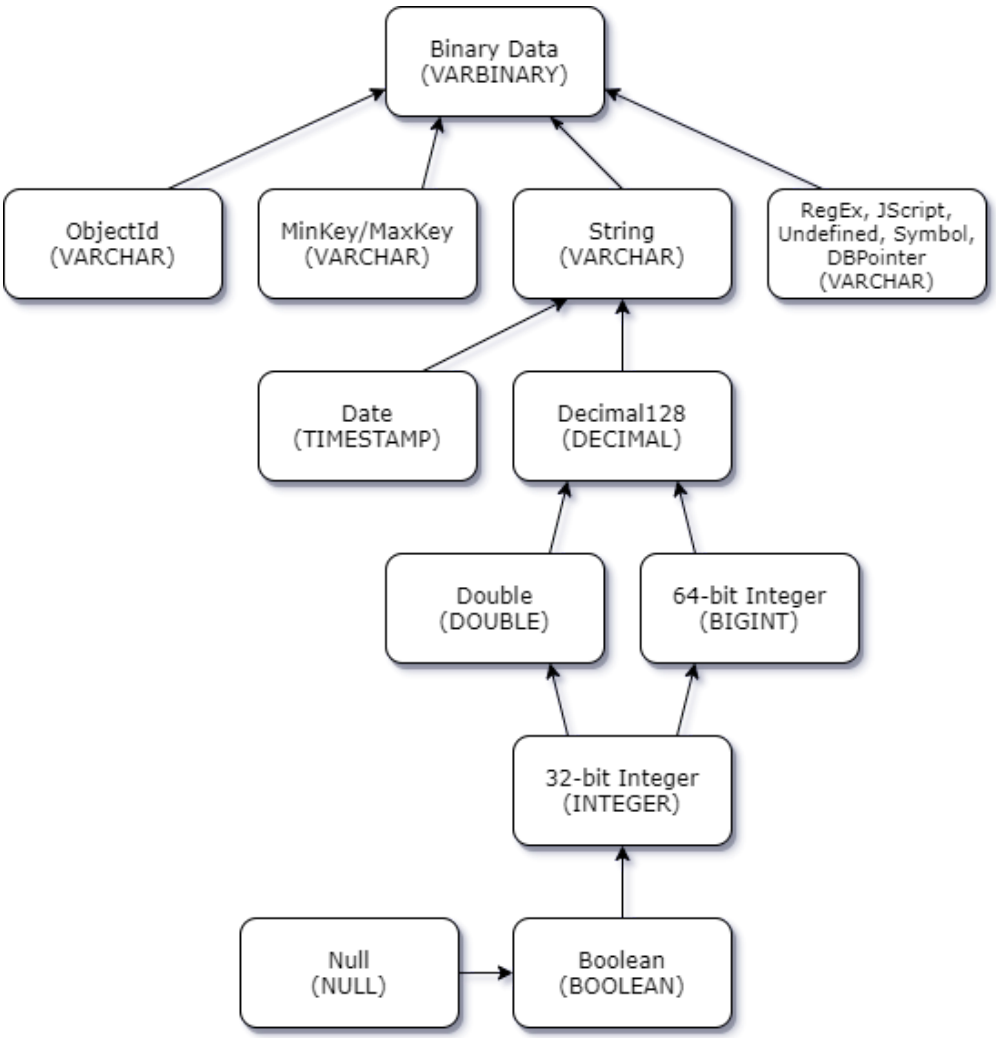
{
"_id" : "112244",
"name" : "Benjamin",
"age" : "32"
}
```

Das Altersfeld hat im ersten Dokument den Typ 32-Bit-Ganzzahl, im zweiten Dokument jedoch eine Zeichenfolge. Hier stuft der JDBC-Treiber den JDBC-Datentyp auf VARCHAR um, um beide Datentypen zu verarbeiten, wenn er angetroffen wird.

Tabellenname	Spaltenname	Datentyp	Key (Schlüssel)
Team	Team-ID	VARCHAR	PK
Team	Name	VARCHAR	
Team	age	VARCHAR	

Scalar-scalar Förderung von Konflikten

Das folgende Diagramm zeigt, wie Konflikte zwischen skalaren Datentypen gelöst werden.



Scalar-complex Geben Sie Konfliktherstellung ein

Wie bei Konflikten zwischen skalarem Typ und Skalar kann dasselbe Feld in verschiedenen Dokumenten widersprüchliche Datentypen zwischen komplex (Array und Objekt) und skalar (Integer, Boolean usw.) aufweisen. Alle diese Konflikte werden für diese Felder in VARCHAR aufgelöst (heraufgestuft). In diesem Fall werden Array- und Objektdaten als JSON-Repräsentation zurückgegeben.

Beispiel für einen Konflikt zwischen eingebettetem Array und Zeichenfolgenfeld:

```
{
   "_id":"112233",
   "name":"George Jackson",
   "subscriptions":[
      "Vogue",
      "People",
      "USA Today"
   ]
}
{
   "_id":"112244",
   "name":"Joan Starr",
   "subscriptions":1
}
```

Das vorherige Beispiel ist dem Schema für die Tabelle customer2 zugeordnet:

Tabellenname	Spaltenname	Datentyp	Key (Schlüssel)
Kunde2	Kunde2 ist	VARCHAR	PK
Kunde2	Name	VARCHAR	
Kunde2	Abonnement	VARCHAR	

und die virtuelle Tabelle customer1_subscriptions:

Tabellenname	Spaltenname	Datentyp	Key (Schlüssel)
customer1_subscriptions	Kunde 1 ist	VARCHAR	PK/FK
customer1_Abonnements	abonnement_index_lvl0	BIGINT	PK
Kunde 1: Abonnements	value	VARCHAR	
customer_address	city	VARCHAR	
customer_address	Region	VARCHAR	
customer_address	country	VARCHAR	
customer_address	Code	VARCHAR	

Behandlung von Objekt- und Array-Datentypen

Bisher haben wir nur beschrieben, wie skalare Datentypen abgebildet werden. Objekt- und Array-Datentypen werden (derzeit) virtuellen Tabellen zugeordnet. Der JDBC-Treiber erstellt eine virtuelle Tabelle, die entweder Objekt- oder Array-Felder in einem Dokument darstellt. Der Name der zugewiesenen virtuellen Tabelle verkettet den Namen der ursprünglichen Sammlung, gefolgt vom Feldnamen, der durch einen Unterstrich („_“) getrennt ist.

Der Primärschlüssel der Basistabelle („_id“) nimmt in der neuen virtuellen Tabelle einen neuen Namen an und wird als Fremdschlüssel für die zugehörige Basistabelle bereitgestellt.

Für Felder vom Typ eingebettetes Array werden Indexspalten generiert, um den Index im Array auf jeder Ebene des Arrays darzustellen.

Beispiel für ein eingebettetes Objektfeld

Für Objektfelder in einem Dokument wird vom JDBC-Treiber eine Zuordnung zu einer virtuellen Tabelle erstellt.

```
{
```

```
   "Collection: customer",
   "_id":"112233",
   "name":"George Jackson",
   "address":{
      "address1":"123 Avenue Way",
      "address2":"Apt. 5",
      "city":"Hollywood",
      "region":"California",
      "country":"USA",
      "code":"90210"
   }
}
```

Das vorherige Beispiel ist dem Schema für die Kundentabelle zugeordnet:

Tabellenname	Spaltenname	Datentyp	Key (Schlüssel)
customer	Kunden-ID	VARCHAR	PK
customer	Name	VARCHAR	

und die virtuelle Tabelle customer_address:

Tabellenname	Spaltenname	Datentyp	Key (Schlüssel)
customer_address	Kunden-ID	VARCHAR	PK/FK
customer_address	Adresse1	VARCHAR	
customer_address	Adresse2	VARCHAR	
customer_address	city	VARCHAR	
customer_address	Region	VARCHAR	
customer_address	country	VARCHAR	
customer_address	Code	VARCHAR	

Beispiel für ein eingebettetes Array-Feld

Für Array-Felder in einem Dokument wird vom JDBC-Treiber auch eine Zuordnung zu einer virtuellen Tabelle erstellt.

```
{
   "Collection: customer1",
   "_id":"112233",
   "name":"George Jackson",
   "subscriptions":[
      "Vogue",
      "People",
      "USA Today"
   ]
}
```

Das vorherige Beispiel ist dem Schema für die Tabelle customer1 zugeordnet:

Tabellenname	Spaltenname	Datentyp	Key (Schlüssel)
Kunde1	Kunde1 ist	VARCHAR	PK
Kunde1	Name	VARCHAR	

und die virtuelle Tabelle customer1_subscriptions:

Tabellenname	Spaltenname	Datentyp	Key (Schlüssel)
customer1_subscriptions	Kunde 1 ist	VARCHAR	PK/FK
customer1_Abonnements	abonnement_index_lvl0	BIGINT	PK
Kunde 1: Abonnements	value	VARCHAR	
customer_address	city	VARCHAR	
customer_address	Region	VARCHAR	

Tabellenname	Spaltenname	Datentyp	Key (Schlüssel)
customer_address	country	VARCHAR	
customer_address	Code	VARCHAR	

SQL-Unterstützung und Einschränkungen

Der Amazon DocumentDB-JDBC-Treiber ist ein schreibgeschützter Treiber, der eine Teilmenge und einige gängige Erweiterungen unterstützt. SQL-92 Weitere Informationen finden Sie in der Dokumentation zu SQL-Einschränkungen und in der Dokumentation zu den JDBC-Einschränkungen.

Fehlerbehebung

Wenn Sie Probleme bei der Verwendung des Amazon DocumentDB-JDBC-Treibers haben, lesen Sie im Leitfaden zur Fehlerbehebung nach. https://github.com/aws/amazon-documentdb-jdbc-driver/blob/develop/src/markdown/support/troubleshooting-guide.md

Stellen Sie mithilfe des Amazon DocumentDB-ODBC-Treibers eine Verbindung her

Der ODBC-Treiber für Amazon DocumentDB bietet eine SQL-relational Schnittstelle für Entwickler und ermöglicht die Konnektivität von BI-Tools wie Power BI Desktop und Microsoft Excel.

Weitere Informationen finden Sie in der Amazon DocumentDB-ODBC-Treiberdokumentation unter. GitHub

Topics

- Erste Schritte
- Einrichtung des Amazon DocumentDB-ODBC-Treibers in Windows
- Stellen Sie von Microsoft Excel aus eine Verbindung zu Amazon DocumentDB her
- Stellen Sie von Microsoft Power BI Desktop aus eine Verbindung zu Amazon DocumentDB her
- Automatische Schemagenerierung
- SQL-Unterstützung und Einschränkungen
- Fehlerbehebung

Erste Schritte

Schritt 1. Erstellen Sie Amazon DocumentDB-Cluster

Wenn Sie noch keinen Amazon DocumentDB-Cluster haben, gibt es eine Reihe von Möglichkeiten, um loszulegen.

> **Note**
>
> Amazon DocumentDB ist ein Dienst, der nur für Virtual Private Cloud (VPC) verfügbar ist. Wenn Sie von einem lokalen Computer außerhalb der VPC des Clusters aus eine Verbindung herstellen, müssen Sie eine SSH-Verbindung zu einer Amazon EC2-Instance herstellen. Weitere Informationen finden Sie unter Stellen Sie mithilfe von Amazon EC2 eine Verbindung her. Weitere Informationen zum SSH-Tunneling finden Sie unter Verwenden eines SSH-Tunnels zum Herstellen einer Verbindung zu Amazon DocumentDB.

Schritt 2. JRE- oder JDK-Installation

Abhängig von Ihrer BI-Anwendung müssen Sie möglicherweise sicherstellen, dass auf Ihrem Computer eine 64-Bit-JRE- oder JDK-Installation der Version 8 oder höher installiert ist. Sie können das Java SE Runtime Environment 8 hier herunterladen.

Schritt 3. Laden Sie den Amazon DocumentDB ODBC-Treiber herunter

Laden Sie den Amazon DocumentDB ODBC-Treiber hier herunter. https://github.com/aws/amazon-documentdb-odbc-driver/releases Wählen Sie das richtige Installationsprogramm (z. B. documentdb-odbc-1.0.0.msi). Folgen Sie der Installationsanleitung.

Schritt 4. Verwenden Sie einen SSH-Tunnel, um eine Verbindung zu Amazon DocumentDB herzustellen

Amazon DocumentDB-Cluster werden in einer Amazon Virtual Private Cloud (Amazon VPC) bereitgestellt. Auf sie kann direkt von Amazon EC2-Instances oder anderen AWS Diensten zugegriffen werden, die in derselben Amazon VPC bereitgestellt werden. Darüber hinaus können Amazon EC2-Instances oder andere AWS Dienste in verschiedenen VPCs in derselben AWS-Region oder anderen Regionen über VPC-Peering auf Amazon DocumentDB zugreifen.

Nehmen wir jedoch an, Ihr Anwendungsfall erfordert, dass Sie (oder Ihre Anwendung) von außerhalb der VPC des Clusters auf Ihre Amazon DocumentDB-Ressourcen zugreifen. Dies wird

bei den meisten Benutzern der Fall sein, die ihre Anwendung nicht auf einer VM in derselben VPC wie der Amazon DocumentDB-Cluster ausführen. Wenn Sie eine Verbindung von außerhalb der VPC herstellen, können Sie SSH-Tunneling (auch bekannt als Portweiterleitung) verwenden, um auf Ihre Amazon DocumentDB-Ressourcen zuzugreifen.

Um einen SSH-Tunnel zu erstellen, benötigen Sie eine Amazon EC2-Instance, die in derselben Amazon VPC wie Ihr Amazon DocumentDB-Cluster ausgeführt wird. Sie können entweder eine vorhandene EC2-Instance in derselben VPC wie Ihr Cluster verwenden oder eine erstellen. Sie können einen SSH-Tunnel zum Amazon DocumentDB-Cluster einrichten, `sample-cluster.node.us-east-1.docdb.amazonaws.com` indem Sie den folgenden Befehl auf Ihrem lokalen Computer ausführen:

```
ssh -i "ec2Access.pem" -L 27017:sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017 ubuntu@ec2-34-229-221-164.compute-1.amazonaws.com -
N
```

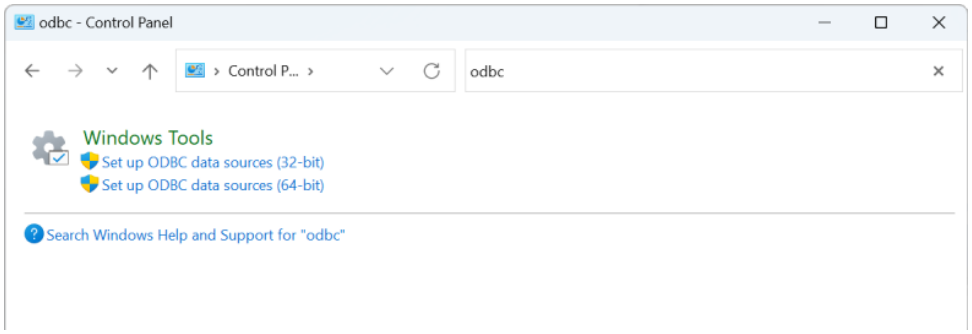
Das `-L`-Flag dient zur Weiterleitung eines lokalen Ports. Dies ist eine Voraussetzung für die Verbindung zu einem BI-Tool, das auf einem Client außerhalb Ihrer VPC ausgeführt wird. Sobald Sie den obigen Schritt ausgeführt haben, können Sie mit den nächsten Schritten für das BI-Tool Ihrer Wahl fortfahren.

Weitere Informationen zum SSH-Tunneling finden Sie in der Dokumentation zur Verwendung eines SSH-Tunnels zur Verbindung mit Amazon DocumentDB.

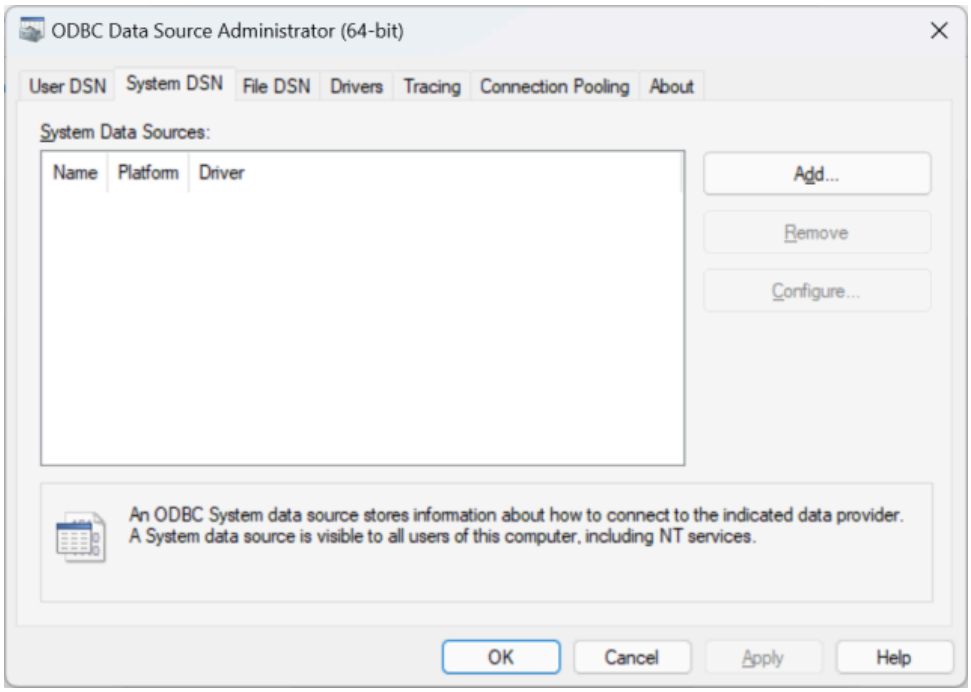
Einrichtung des Amazon DocumentDB-ODBC-Treibers in Windows

Gehen Sie wie folgt vor, um den Amazon DocumentDB-ODBC-Treiber in Windows einzurichten:

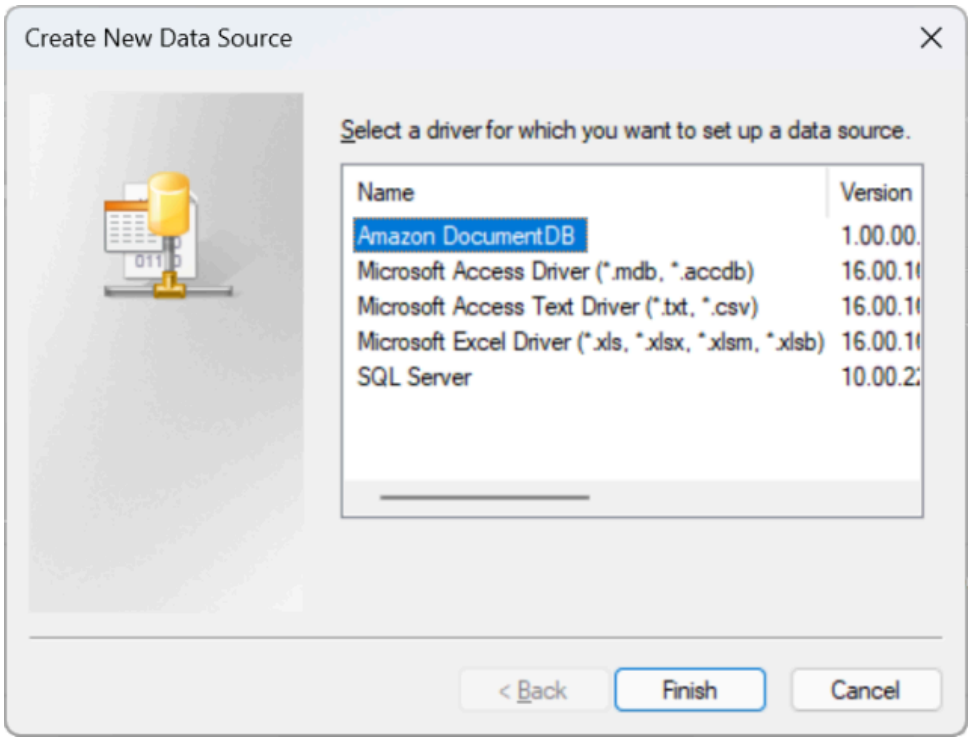
1. Öffnen Sie die Systemsteuerung in Windows und suchen Sie nach ODBC (oder wählen Sie im Menü Windows-Tools > ODBC-Datenquellen (32-Bit) oder ODBC-Datenquellen (64-Bit)):



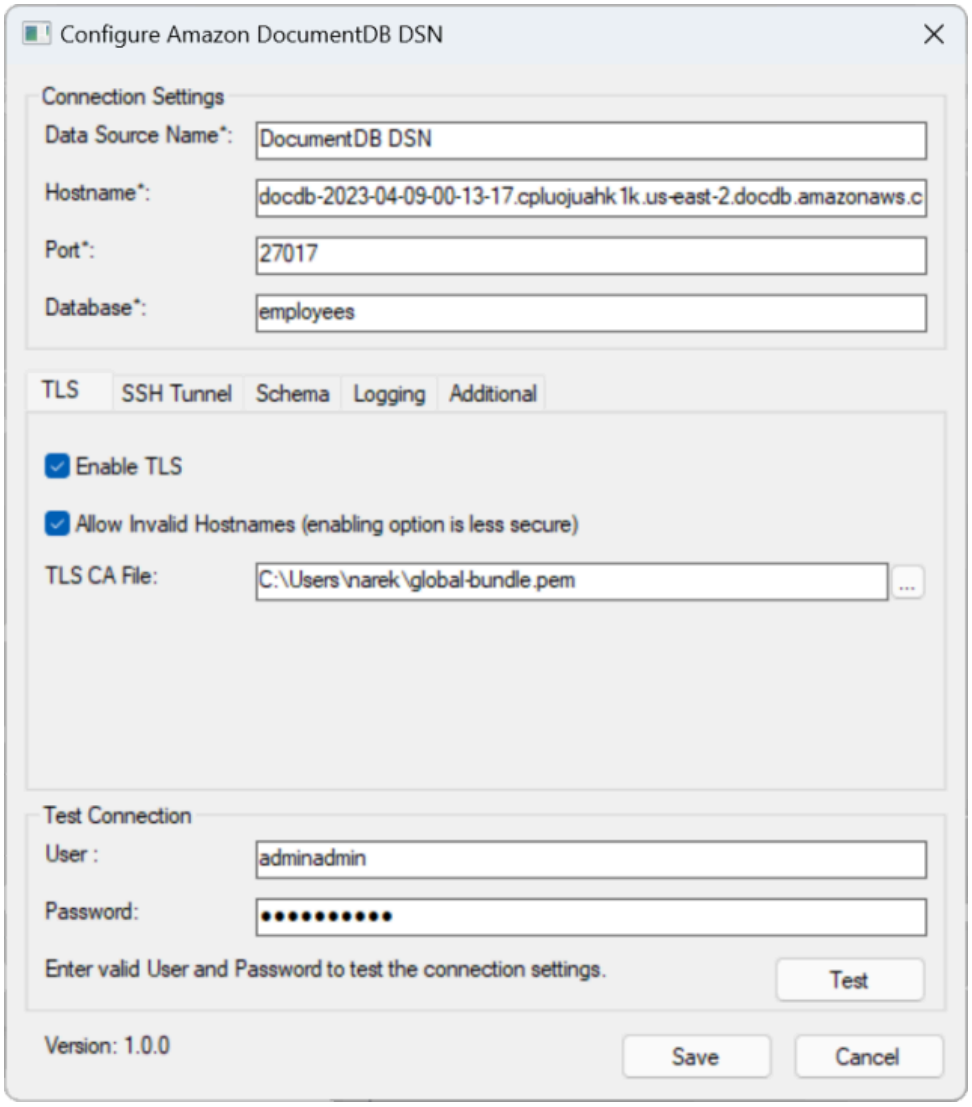
2. Wählen Sie den entsprechenden ODBC-Treiber-Datenquellenadministrator aus: Wählen Sie die 32-Bit-Version, falls sie installiert ist, andernfalls wählen Sie die 64-Bit-Version.
3. Wählen Sie die Registerkarte System-DSN aus und wählen Sie dann Hinzufügen... um einen neuen DSN hinzuzufügen:



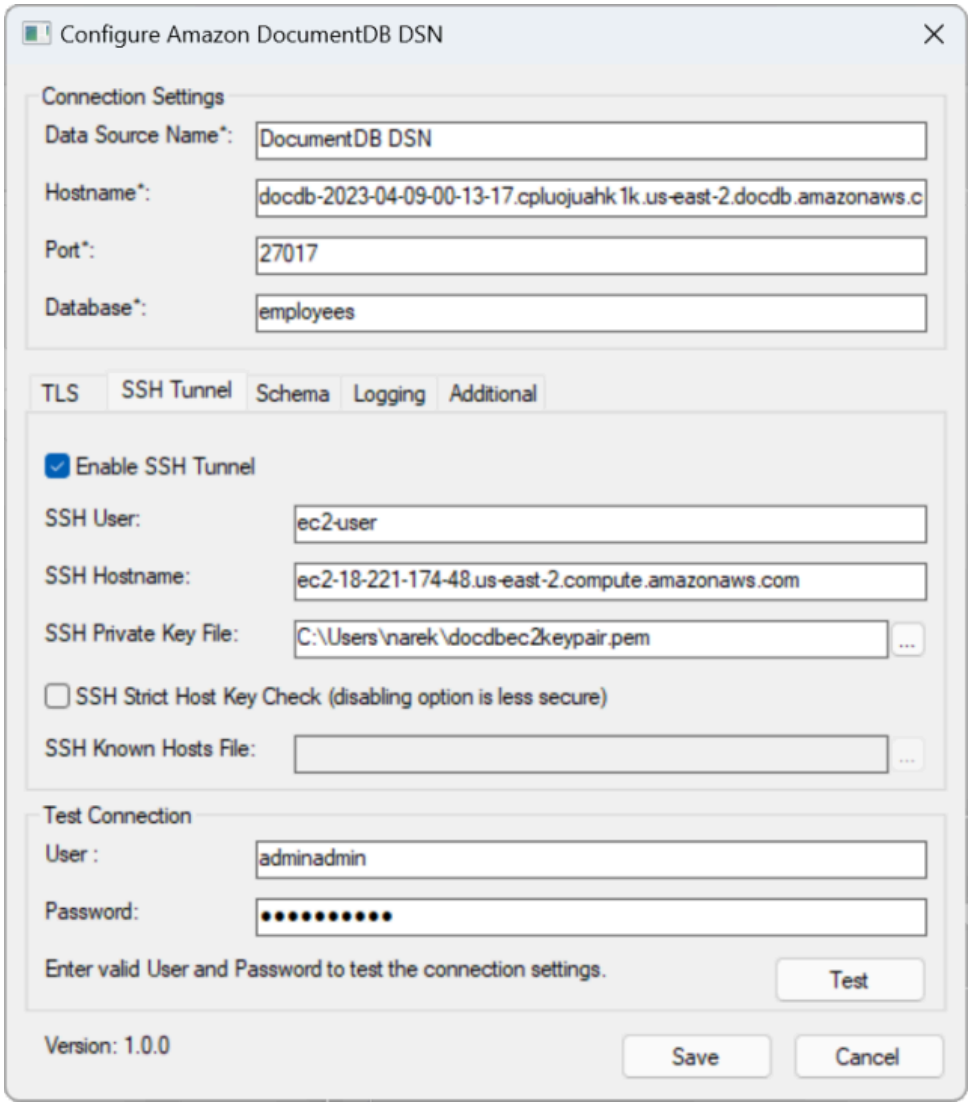
4. Wählen Sie Amazon DocumentDB aus der Liste der Datenquellentreiber aus:



5. Füllen Sie im Dialogfeld „Amazon DocumentDB DSN konfigurieren“ die Felder „Verbindungseinstellungen“, „TLS“ und „Verbindung testen“ aus und wählen Sie dann Speichern aus:



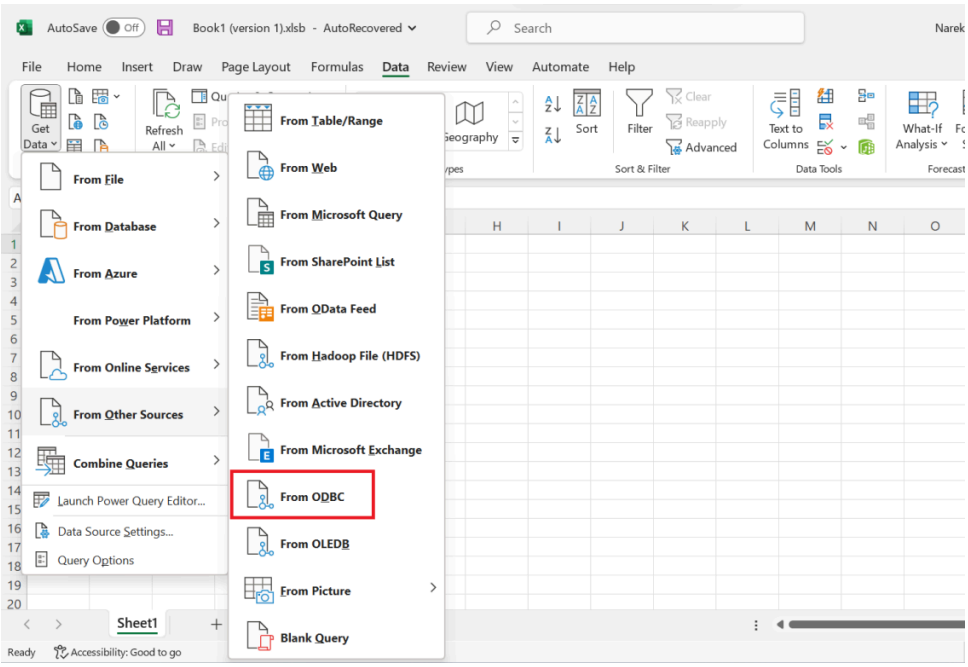
6. Stellen Sie sicher, dass Sie das Windows-Formular korrekt ausfüllen, da sich die Verbindungsdetails je nach der von Ihnen gewählten SSH-Tunneling-Methode zur EC2-Instance unterscheiden. Weitere SSH-Tunneling-Methoden finden Sie hier. https://github.com/aws/amazon-documentdb-odbc-driver/blob/develop/src/markdown/setup/setup.md#using-an-ssh-tunnel-to-connect-to-amazon-documentdb Weitere Informationen zu den einzelnen Eigenschaften finden Sie unter Syntax und Optionen für Verbindungszeichenfolgen.



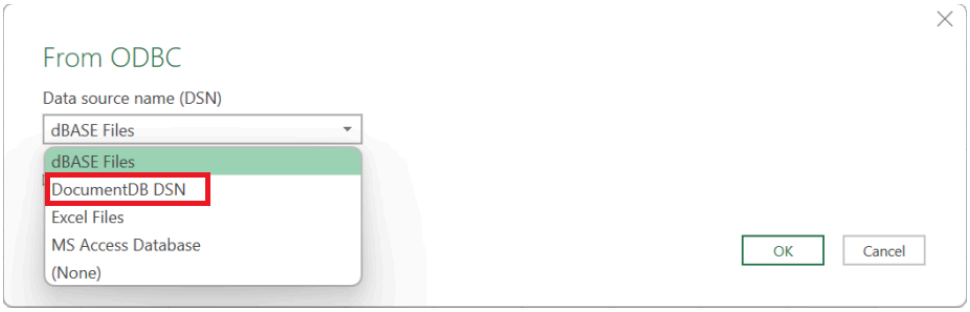
Weitere Informationen zur Konfiguration des Amazon DocumentDB-ODBC-Treibers unter Windows finden Sie hier. https://github.com/aws/amazon-documentdb-odbc-driver/blob/develop/src/markdown/setup/windows-dsn-configuration.md

Stellen Sie von Microsoft Excel aus eine Verbindung zu Amazon DocumentDB her

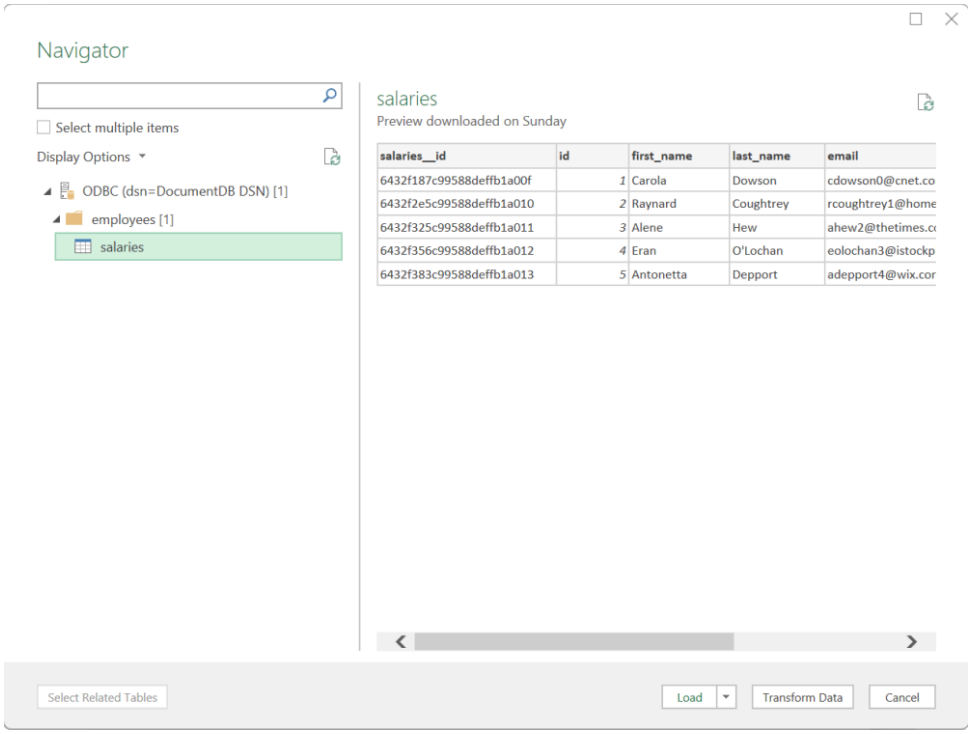
1. Stellen Sie sicher, dass der Amazon DocumentDB-Treiber korrekt installiert und konfiguriert wurde. Weitere Informationen finden Sie unter Einrichtung des Amazon DocumentDB-ODBC-Treibers in Windows.
2. Starten Sie Microsoft Excel.
3. Navigieren Sie zu Daten > Daten abrufen > Aus anderen Quellen.
4. Wählen Sie „Aus ODBC“:



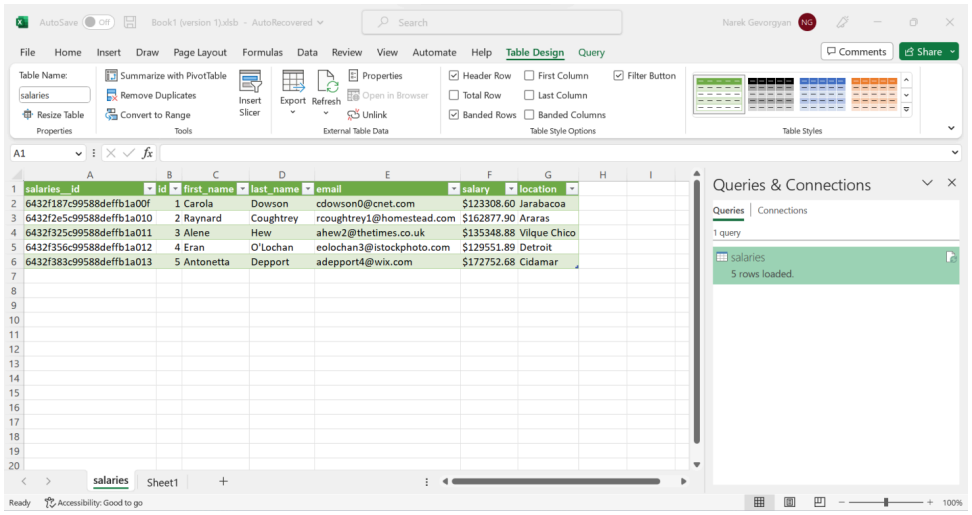
5. Wählen Sie im Dropdownmenü Datenquellenname (DSN) die Datenquelle aus, die Amazon DocumentDB zugeordnet ist:



6. Wählen Sie die Sammlung aus, aus der Sie Daten in Excel laden möchten:



7. Daten in Excel laden:



Stellen Sie von Microsoft Power BI Desktop aus eine Verbindung zu Amazon DocumentDB her

Topics

- Voraussetzungen
- Hinzufügen eines benutzerdefinierten Connectors für Microsoft Power BI Desktop
- Verbindung mithilfe des benutzerdefinierten Amazon DocumentDB-Connectors herstellen

- Konfiguration von Microsoft Power BI Gateway

Voraussetzungen

Bevor Sie beginnen, stellen Sie sicher, dass der Amazon DocumentDB-ODBC-Treiber korrekt installiert ist.

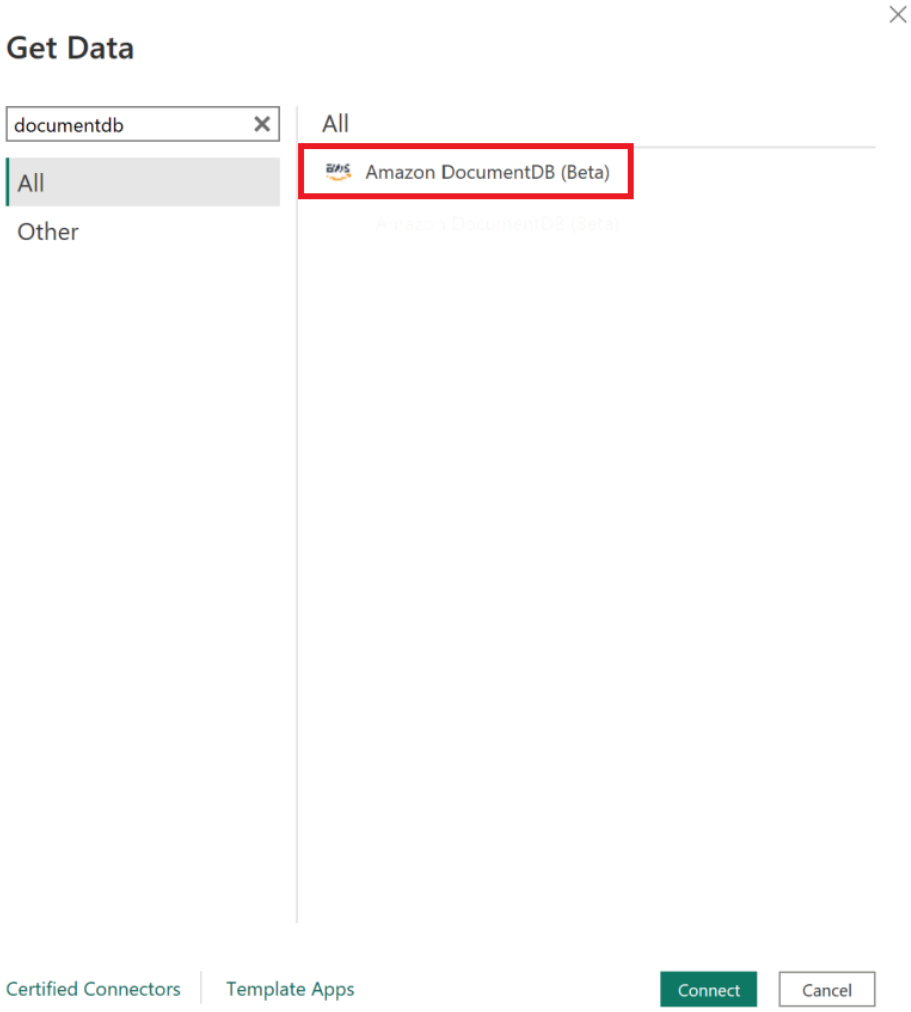
Hinzufügen eines benutzerdefinierten Connectors für Microsoft Power BI Desktop

Kopieren Sie die `AmazonDocumentDBConnector.mez` Datei in den `<User>\Documents \Power BI Desktop\Custom Connectors\` Ordner (oder in den Ordner, `<User>\OneDrive \Documents\Power BI Desktop\Custom Connectors` falls Sie ihn verwenden OneDrive). Dadurch kann Power BI auf den benutzerdefinierten Connector zugreifen. Den Connector für Power BI Desktop finden Sie hier. Starten Sie Power BI Desktop neu, um sicherzustellen, dass der Connector geladen ist.

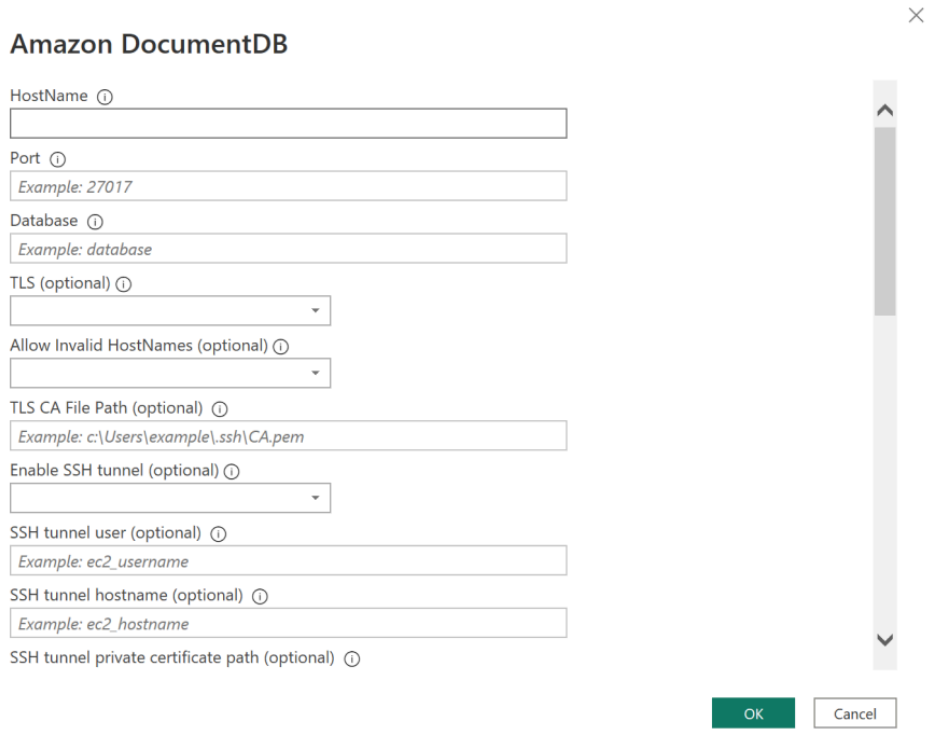
> **Note**
>
> Der benutzerdefinierte Connector unterstützt nur den Amazon DocumentDB-Benutzernamen und das Passwort für die Authentifizierung.

Verbindung mithilfe des benutzerdefinierten Amazon DocumentDB-Connectors herstellen

1. Wählen Sie Amazon DocumentDB (Beta) unter Get Data aus und wählen Sie Connect. Wenn Sie wegen der Nutzung eines Drittanbieterdienstes eine Warnung erhalten, wählen Sie Weiter.



2. Geben Sie alle erforderlichen Informationen ein, um eine Verbindung zu Ihrem Amazon DocumentDB-Cluster herzustellen, und wählen Sie dann OK:



> **Note**
>
> Je nach Konfiguration des Datenquellennamens (DSN) Ihres ODBC-Treibers wird der Bildschirm mit den SSH-Verbindungsdetails möglicherweise nicht angezeigt, wenn Sie die erforderlichen Informationen bereits in den DSN-Einstellungen angegeben haben.

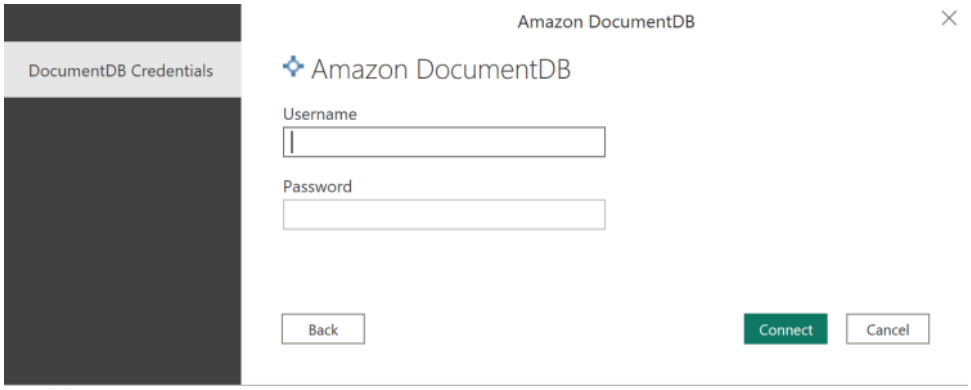
3. Wählen Sie den Datenverbindungsmodus:

   - Import — lädt alle Daten und speichert die Informationen auf der Festplatte. Die Daten müssen aktualisiert und neu geladen werden, damit Datenaktualisierungen angezeigt werden.
   - Direktabfrage — lädt keine Daten, sondern führt Live-Abfragen der Daten durch. Das bedeutet, dass Daten nicht aktualisiert und neu geladen werden müssen, um Datenaktualisierungen anzuzeigen.

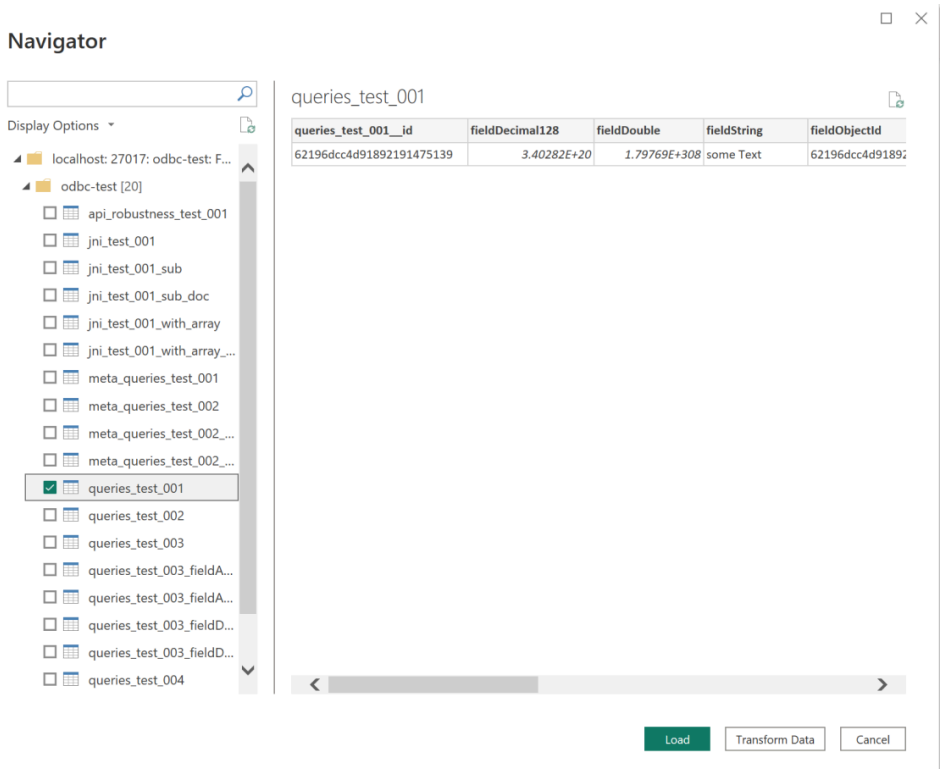


> **Note**
>
> Wenn Sie einen sehr großen Datensatz verwenden, kann das Importieren aller Daten einen längeren Zeitraum in Anspruch nehmen.

4. Wenn Sie zum ersten Mal eine Verbindung zu dieser Datenquelle herstellen, wählen Sie den Authentifizierungstyp aus und geben Sie Ihre Anmeldeinformationen ein, wenn Sie dazu aufgefordert werden. Wählen Sie dann Verbinden:



5. Wählen Sie im Navigator-Dialogfeld die gewünschten Datenbanktabellen aus und wählen Sie dann entweder „Laden“, um die Daten zu laden, oder „Daten transformieren“, um die Transformation der Daten fortzusetzen.



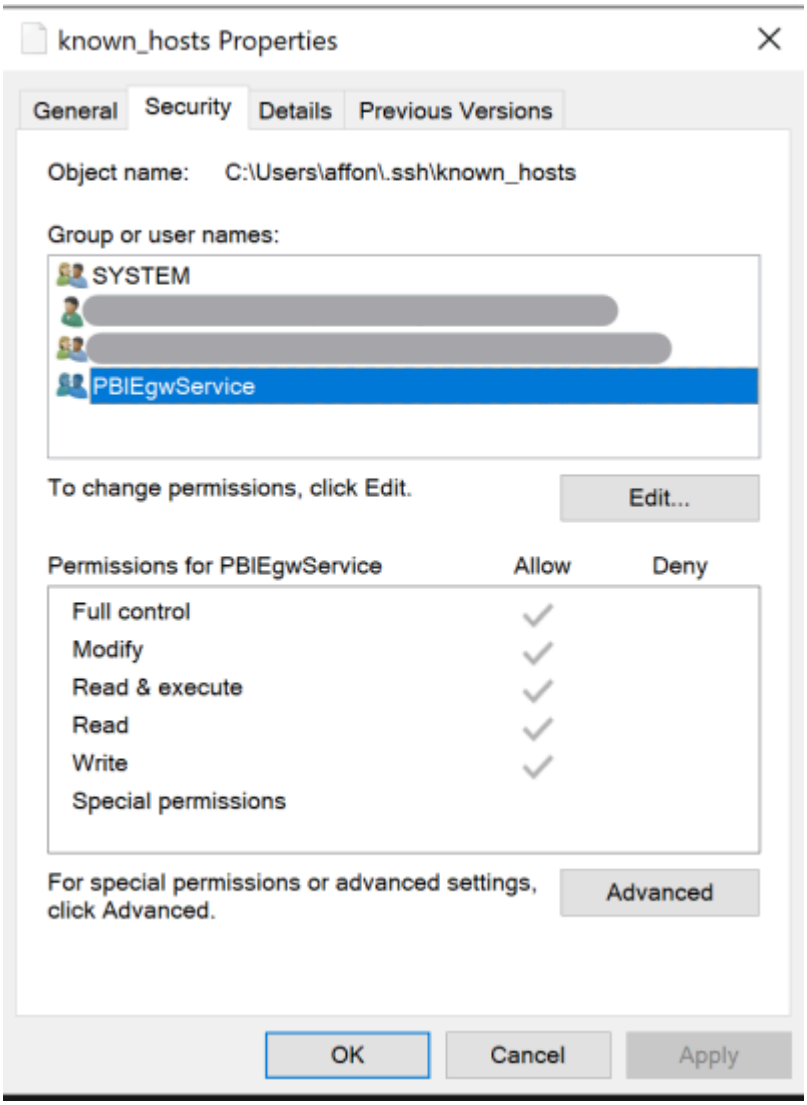
> **Note**
>
> Ihre Datenquelleneinstellungen werden gespeichert, sobald Sie eine Verbindung herstellen. Um sie zu ändern, wählen Sie „Daten transformieren“ > „Datenquelleneinstellungen“.

Konfiguration von Microsoft Power BI Gateway

Voraussetzungen:

- Stellen Sie sicher, dass der benutzerdefinierte Connector mit Power BI Gateway funktioniert.
- Stellen Sie sicher, dass der ODBC-DSN in den ODBC-Datenquellen auf der Registerkarte System auf dem Computer erstellt wird, auf dem Power BI Gateway installiert ist.

Wenn Sie die interne SSH-Tunnelfunktion verwenden, `known_hosts` muss sich die Datei dort befinden, wo das Power BI-Dienstkonto Zugriff darauf hat.



> **Note**
>
> Dies gilt auch für alle Dateien, die Sie möglicherweise benötigen, um eine Verbindung zu Ihrem Amazon DocumentDB-Cluster herzustellen, z. B. eine Zertifizierungsstelle (CA) - Zertifikatsdatei (PEM-Datei).

Automatische Schemagenerierung

Der ODBC-Treiber verwendet den Amazon DocumentDB-JDBC-Treiber über JNI (Java Native Interface), sodass die Funktion zur automatischen Schemagenerierung im JDBC-Treiber ähnlich funktioniert. Weitere Informationen zur automatischen Schemagenerierung finden Sie

unterAutomatische JDBC-Schemagenerierung. Weitere Informationen zur ODBC-Treiberarchitektur finden Sie unter Amazon DocumentDB ODBC-Treiberarchitektur unter. GitHub

SQL-Unterstützung und Einschränkungen

Der Amazon DocumentDB-ODBC-Treiber ist ein schreibgeschützter Treiber, der eine Teilmenge SQL-92 und einige gängige Erweiterungen unterstützt. Weitere Informationen finden Sie in der Dokumentation zur ODBC-Unterstützung und zu den Einschränkungen.

Fehlerbehebung

Wenn Sie Probleme bei der Verwendung des Amazon DocumentDB-ODBC-Treibers haben, lesen Sie im Leitfaden zur Fehlerbehebung nach.

Programm mit Amazon DocumentDB

Der Dienst unterstützt die JSON-Schemavalidierung.

Themen

- Amazon DocumentDB Java-Programmierhandbuch
- Verwenden der JSON-Schemavalidierung

Amazon DocumentDB Java-Programmierhandbuch

Dieses umfassende Handbuch bietet eine detaillierte Anleitung zur Arbeit mit Amazon DocumentDB mithilfe der Java-Treiber von MongoDB und behandelt wichtige Aspekte des Datenbankbetriebs und der Datenbankverwaltung.

Themen

- Einführung
- Voraussetzungen
- Datenmodelle für diesen Leitfaden
- Stellen Sie mit einem MongoDB-Java-Treiber eine Verbindung zu Amazon DocumentDB her
- Durchführung von CRUD-Operationen in Amazon DocumentDB mit Java
- Indexverwaltung in Amazon DocumentDB mit Java

- Event-driven Programmierung mit Amazon DocumentDB und Java

Einführung

Das Handbuch beginnt mit der Konnektivität und erklärt, wie Sie mithilfe des MongoDB-Java-Treibers sichere Verbindungen zu DocumentDB-Clustern herstellen. Es beschreibt die Komponenten der Verbindungszeichenfolge, die SSL/TLS Implementierung und verschiedene Verbindungsoptionen, einschließlich IAM-Authentifizierung und Verbindungspooling, sowie robuste Strategien zur Fehlerbehandlung.

Im Abschnitt CRUD-Operationen (Create, Read, Update, Delete) behandelt das Handbuch ausführlich die Bearbeitung von Dokumenten und zeigt, wie Dokumente sowohl mit Einzel- als auch mit Massenoperationen erstellt, gelesen, aktualisiert und gelöscht werden. Es erklärt die Verwendung von Filtern, Abfragen und verschiedenen Operationsoptionen und betont gleichzeitig bewährte Methoden für die Fehlerbehandlung und die Implementierung einer Wiederholungslogik zur Verbesserung der Zuverlässigkeit. Das Handbuch behandelt auch ausführlich die Indexverwaltung und beschreibt detailliert die Erstellung und Pflege verschiedener Indextypen, einschließlich Einzelfeld-, zusammengesetzter, dünn besetzter Indizes und Textindizes. Es wird erklärt, wie die Abfrageleistung durch die richtige Indexauswahl und Verwendung der `explain()` Funktion zur Analyse von Abfrageausführungsplänen optimiert werden kann.

Der letzte Abschnitt konzentriert sich auf die ereignisgesteuerte Programmierung mithilfe der Change Streams von Amazon DocumentDB und zeigt, wie die Überwachung von Datenänderungen in Echtzeit in Java-Anwendungen implementiert wird. Es behandelt die Implementierung von Change-Stream-Cursoren, die Handhabung von Resume-Tokens für den kontinuierlichen Betrieb und zeitbasierte Operationen für die historische Datenverarbeitung. Der Leitfaden enthält durchgehend praktische Codebeispiele und bewährte Methoden, was ihn zu einer unschätzbaren Ressource für Sie macht, wenn Sie robuste Java-Anwendungen mit Amazon DocumentDB erstellen.

Voraussetzungen

Stellen Sie vor dem Beginn sicher, dass Sie über das Folgende verfügen:

- Und AWS-Konto mit einem konfigurierten Amazon DocumentDB-Cluster. Informationen zur Clustereinrichtung finden Sie im Datenbank-Blog unter Erste Schritte mit Amazon DocumentDB mithilfe von Amazon EC2. AWS
- Java Development Kit (JDK) ist installiert (wir werden Amazon Corretto 21 für dieses Handbuch verwenden).

- Maven für das Abhängigkeitsmanagement.

Datenmodelle für diesen Leitfaden

Der gesamte Beispielcode in diesem Handbuch geht von einer Verbindung zu einer „ProgGuideData“ -Testdatenbank aus, die eine Sammlung „Restaurants“ enthält. Alle Beispielcodes in diesem Handbuch funktionieren auf einem System mit Restaurantlisten. Im Folgenden finden Sie ein Beispiel dafür, wie ein Dokument in diesem System aussieht:

```
{
    "_id": "ab6ad8f119b5bca3efa2c7ae",
    "restaurantId": "REST-CRT9BL",
    "name": "Thai Curry Palace",
    "description": "Amazing Restaurant, must visit",
    "cuisine": "Thai",
    "address": {
        "street": "914 Park Street",
        "city": "Bryan",
        "state": "AL",
        "zipCode": "96865",
        "location": {
            "type": "Point",
            "coordinates": [-25.4619, 8.389]
        }
    },
    "contact": {
        "phone": "(669) 915-9056 x6657"
    },
    "rating": {
        "average": 3.4,
        "totalReviews": 275
    },
    "priceRange": "$",
    "menu": [{
        "category": "Appetizers",
        "items": [{
            "name": "Buffalo Chicken Wings",
            "price": 13.42
        }]
    }],
    "features": [
        "Private Dining"
```

```
    ],
    "isActive": false“ michelin”: {“
        star”: 3,
        “ranking_years”: 4
    }
}
```

Bei allen Codebeispielen, die CRUD, Indexverwaltung und ereignisgesteuerte Programmierung zeigen, wird davon ausgegangen, dass Sie ein `MongoClient` Objekt`dbClient`, ein Objekt und ein `MongoDatabase` Objekt `connection`DB haben. `MongoCollection` `collection`

> **Note**
>
> Alle Codebeispiele in diesem Handbuch wurden mit der MongoDB-Java-Treiberversion 5.3.0 getestet.

Stellen Sie mit einem MongoDB-Java-Treiber eine Verbindung zu Amazon DocumentDB her

Dieser Abschnitt enthält eine schrittweise Anleitung zum Herstellen einer Verbindung zu Amazon DocumentDB mithilfe von Java-Treibern. Dies wird Ihnen den Einstieg in die Integration von DocumentDB in Ihre Java-Anwendungen erleichtern.

Themen

- Schritt 1: Einrichten des Projekts
- Schritt 2: Erstellen Sie die Verbindungszeichenfolge
- Schritt 3: Schreiben Sie den Verbindungscode
- Schritt 4: Verbindungsausnahmen behandeln
- Schritt 5: Den Code ausführen
- Bewährte Methoden für Verbindungen

Schritt 1: Einrichten des Projekts

1. Erstellen Sie mit Maven ein Java-Projekt:

```
mvn archetype:generate -DgroupId=com.docdb.guide -DartifactId=my-docdb-project -
DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

2. Fügen Sie den MongoDB-Java-Treiber als Abhängigkeit für das Projekt in Ihrer Datei 'pom.xml' hinzu:

```
<dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver-sync</artifactId>
    <version>5.3.0</version>
</dependency>
```

Schritt 2: Erstellen Sie die Verbindungszeichenfolge

Die Amazon DocumentDB-Verbindungszeichenfolge ist wichtig, um eine Verbindung zwischen Ihrer Anwendung und Ihrem DocumentDB-Cluster herzustellen. Diese Zeichenfolge enthält wichtige Informationen wie Cluster-Endpunkt, Port, Authentifizierungsdetails und verschiedene Verbindungsoptionen. Um eine DocumentDB-Verbindungszeichenfolge zu erstellen, beginnen Sie normalerweise mit dem Basisformat:

```
"mongodb://username:password@cluster-endpoint:port/?[connection options]"
```

Sie müssen „Benutzername“ und „Passwort“ durch Ihre tatsächlichen Anmeldeinformationen ersetzen. Den Endpunkt und die Portnummer Ihres Clusters finden Sie sowohl im AWS-Managementkonsole als auch über das AWS CLI. Informationen Die Endpunkte eines Clusters finden zum Finden des Cluster-Endpunkts für Ihren Cluster finden Sie unter. Der Standardport für DocumentDB ist 27017.

Beispiele für Verbindungszeichenfolgen

- Herstellen einer Verbindung zu DocumentDB mithilfe von Verschlüsselung während der Übertragung und Vergewissern Sie sich, dass Leseanforderungen an Read Replicas weitergeleitet und in die Primärdatei geschrieben werden:

```
"mongodb://username:password@cluster-endpoint:27017/?tls=true&
   tlsCAFile=global-bundle.pem&
   readPreference=secondaryPreferred&
   retryWrites=false"
```

- Herstellen einer Verbindung zu DocumentDB mithilfe der IAM-Authentifizierung:

```
"mongodb://cluster-endpoint:27017/?tls=true&
   tlsCAFile=global-bundle.pem&
   readPreference=secondaryPreferred&
   retryWrites=false&
   authSource=%24external&
   authMechanism=MONGODB-AWS"
```

Die verschiedenen verfügbaren Optionen für die Verbindungszeichenfolge lauten wie folgt:

- TLS-Zertifikat
- Aus gelesenen Replikaten lesen
- Schreiben Sie Ihr Anliegen und führen Sie ein Tagebuch
- RetryWrites
- IAM-Authentifizierung
- Verbindungspool
- Parameter für das Verbindungs-Timeout

TLS-Zertifikat

`tls=true|false`— Diese Option aktiviert oder deaktiviert Transport Layer Security (TLS). Standardmäßig ist die Verschlüsselung bei der Übertragung auf dem Amazon DocumentDB-Cluster aktiviert. Daher sollte der Wert für diese Option lauten, sofern TLS nicht auf Cluster-Ebene deaktiviert ist. `true`

Bei Verwendung von TLS muss der Code ein SSL-Zertifikat bereitstellen, wenn eine Verbindung zu einem DocumentDB-Cluster hergestellt wird. Laden Sie das Zertifikat herunter, das für die sichere Verbindung zum Cluster erforderlich ist: `global-bundle.pem`. Es gibt zwei Möglichkeiten, die `global-bundle.pem` Datei zu verwenden.

- Option 1 — Extrahieren Sie alle Zertifikate aus der `global-bundle.pem` Datei und verwenden Sie das Keytool von Java, um sie in einer `.jks` Datei zu speichern, die später im Code verwendet werden kann. Das Skript, das zeigt, wie Verbindung bei aktiviertem TLS herstellen das geht, finden Sie auf der Registerkarte Java.

- Option 2 — Fügen Sie die `global-bundle.pem` Datei dynamisch zum Code hinzu, erstellen Sie einen speicherinternen Keystore und verwenden Sie ihn`SSLContext`, um das Zertifikat als Teil des Verbindungsaufbaus bereitzustellen.

Aus gelesenen Replikaten lesen

`replicaSet=rs0&readPreference=secondaryPreferred`— Wenn Sie diese beiden Optionen angeben, werden alle Leseanforderungen an die Read Replicas und die Schreibanforderungen an die primäre Instance weitergeleitet. Die Verwendung `replicaSet=rs0` in der Verbindungszeichenfolge ermöglicht es dem MongoDB-Treiber, eine automatisch aktualisierte Ansicht der Cluster-Topologie beizubehalten, sodass Anwendungen den Überblick über aktuelle Knotenkonfigurationen behalten können, wenn Instanzen hinzugefügt oder entfernt werden. Wenn diese Optionen nicht bereitgestellt oder nicht angegeben werden, werden alle Lese- und Schreibvorgänge an die primäre Instanz `readPreference=primary` gesendet. Weitere Optionen für finden `readPreference` Sie unterPräferenzoptionen lesen.

Schreiben Sie Ihr Anliegen und führen Sie ein Tagebuch

Das Problem beim Schreiben bestimmt den Grad der Bestätigung, die von der Datenbank für Schreibvorgänge angefordert wird. MongoDB-Treiber bieten eine Option zur Optimierung von Write-Concern- und Journal-Dateien. Amazon DocumentDB erwartet nicht, dass Sie write concern und journal einstellen, und ignoriert die für und `j` (wund) `writeConcern` gesendeten Werte. `journal` DocumentDB schreibt Daten immer mit`writeConcern`: `majority` und`journal`:, `true` sodass die Schreibvorgänge auf den meisten Knoten dauerhaft aufgezeichnet werden, bevor eine Bestätigung an den Client gesendet wird.

RetryWrites

`retryWrites`— Ab der Engine-Version 8.0.2 unterstützt Amazon DocumentDB Schreibversuche, die wiederholt werden können. Auf „Aktivieren“ setzen. `retryWrites=true` Bei Engine-Versionen vor 8.0.2 setzen Sie dieses Attribut auf`false`. Weitere Informationen finden Sie unter Wiederholbare Schreibvorgänge in Amazon DocumentDB.

IAM-Authentifizierung

`authSource=%24external`und **`authMechanism=MONGODB-AWS`** — Diese beiden Parameter werden zur Authentifizierung mit verwendet. AWS Identity and Access Management Die IAM-Authentifizierung ist derzeit nur in der instanzbasierten Cluster-Version 5.0 verfügbar. Weitere Informationen finden Sie unter Authentifizierung mithilfe der IAM-Identität.

Verbindungspool

Diese Optionen sind für das Verbindungspooling verfügbar:

- **`maxPoolSize`**— Legt die maximale Anzahl von Verbindungen fest, die im Pool erstellt werden können. Wenn alle Verbindungen verwendet werden und eine neue Anfrage eingeht, wartet es darauf, dass eine Verbindung verfügbar wird. Die Standardeinstellung für MongoDB-Java-Treiber ist 100.
- **`minPoolSize`**— Gibt die Mindestanzahl von Verbindungen an, die im Pool jederzeit aufrechterhalten werden sollten. Die Standardeinstellung für MongoDB-Java-Treiber ist 0.
- **`maxIdleTimeMS`**— Legt fest, wie lange eine Verbindung im Pool inaktiv bleiben kann, bevor sie geschlossen und entfernt wird. Die Standardeinstellung für MongoDB-Java-Treiber ist 100 Millisekunden.
- **`waitQueueTimeoutMS`**— Konfiguriert, wie lange ein Thread warten soll, bis eine Verbindung verfügbar ist, wenn der Pool seine maximale Größe erreicht hat. Wenn eine Verbindung innerhalb dieser Zeit nicht verfügbar ist, wird eine Ausnahme ausgelöst. Der Standardwert für MongoDB-Java-Treiber ist 120.000 Millisekunden (2 Minuten).

Parameter für das Verbindungs-Timeout

Ein Timeout ist ein Mechanismus, mit dem die Zeit begrenzt wird, die ein Vorgang oder Verbindungsversuch in Anspruch nehmen kann, bevor er als fehlgeschlagen betrachtet wird. Die folgenden Timeout-Parameter sind verfügbar, um unbestimmte Wartezeiten zu verhindern und die Ressourcenzuweisung zu verwalten:

- **`connectTimeoutMS`**— Konfiguriert, wie lange der Treiber wartet, um eine Verbindung zum Cluster herzustellen. Die Standardeinstellung ist 10.000 Millisekunden (10 Sekunden).
- **`socketTimeoutMS`**— Gibt an, wie lange der Treiber bei einem Vorgang ohne Schreibzugriff auf eine Antwort vom Server wartet. Die Standardeinstellung ist 0 (kein Timeout oder unendlich).
- **`serverSelectionTimeoutMS`**— Gibt an, wie lange der Treiber darauf wartet, einen verfügbaren Server im Cluster zu finden. Der Standardwert für diese Einstellung beträgt 30 Sekunden und reicht aus, um während des Failovers eine neue primäre Instance auszuwählen.

Schritt 3: Schreiben Sie den Verbindungscode

Das folgende Codebeispiel zeigt, wie eine TLS-Verbindung zu Amazon DocumentDB hergestellt wird:

- Es erstellt Javas KeyStore und SSLContext > -Objekte.
- Es erstellt das MongoClientSettings Objekt auch, indem es an das ConnectionString Objekt übergeben wird. Um eine TLS-Verbindung herzustellen, müssen Sie das MongoClientSettings Objekt verwenden, um das connectionstring und zu bindensslcontext.
- Using MongoClients ruft ein MongoClient Objekt ab.

```
public static MongoClient makeDbConnection(String dbName, String DbUserName, String
 DbPassword,
    String DbClusterEndPoint, String keyStorePass) throws Exception {
    MongoClient connectedClient;
    String connectionOptions = "?
replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false";
    String connectionUrl = "mongodb://" + DbUserName + ":" + DbPassword + "@" +
 DbClusterEndPoint + ":27017/" +
        dbName + connectionOptions;

    try {
        KeyStore trustStore = KeyStore.getInstance(KeyStore.getDefaultType());
        try (FileInputStream fis = new FileInputStream("src/main/resources/certs/
truststore.jks")) {
            trustStore.load(fis, keyStorePass.toCharArray());
            TrustManagerFactory tmf =
 TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
            tmf.init(trustStore);

            SSLContext sslContext = SSLContext.getInstance("TLSv1.2");
            sslContext.init(null, tmf.getTrustManagers(), new SecureRandom());
            ConnectionString connectionString = new ConnectionString(connectionUrl);
            MongoClientSettings settings = MongoClientSettings.builder()
                .applyConnectionString(connectionString)
                .applyToSslSettings(builder - > {
                    builder.enabled(true);
                    builder.context(sslContext);
                })
                .build();
            connectedClient = MongoClients.create(settings);
        }
        return connectedClient;
    } catch (MongoException e5) {
        throw new RuntimeException(e5);
```

```
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}
```

Schritt 4: Verbindungsausnahmen behandeln

Bei der Arbeit mit DocumentDB in Java-Anwendungen ist der Umgang mit Verbindungsausnahmen entscheidend für die Aufrechterhaltung robuster und zuverlässiger Datenbankoperationen. Bei ordnungsgemäßer Verwaltung helfen diese Ausnahmen nicht nur bei der schnellen Diagnose von Problemen, sondern stellen auch sicher, dass Ihre Anwendung vorübergehende Netzwerkunterbrechungen oder Serverausfälle problemlos bewältigen kann, was zu einer verbesserten Stabilität und Benutzerfreundlichkeit führt. Zu den kritischen Ausnahmen im Zusammenhang mit dem Verbindungsaufbau gehören:

- **`MongoException`**— Eine allgemeine Ausnahme, die in verschiedenen Szenarien gelten könnte, für die keine spezifischeren Ausnahmen gelten. Stellen Sie sicher, dass diese Ausnahme nach allen anderen spezifischen Ausnahmen behandelt wird, da es sich um eine allgemeine Catch-All-MongoDB-Ausnahme handelt.
- **`MongoTimeoutException`**— Wird ausgegeben, wenn bei einem Vorgang ein Timeout auftritt. Beispielsweise das Abfragen eines nicht vorhandenen Cluster-Endpunkts.
- **`MongoSocketException`**— Ausgestellt für Netzwerkprobleme. Zum Beispiel eine plötzliche Netzwerkunterbrechung während einer Operation.
- **`MongoSecurityException`**— Wird ausgegeben, wenn die Authentifizierung fehlschlägt. Zum Beispiel beim Herstellen einer Verbindung mit falschen Anmeldeinformationen.
- **`MongoConfigurationException`**— Wird ausgegeben, wenn ein Fehler in der Client-Konfiguration auftritt. Beispielsweise bei Verwendung einer ungültigen Verbindungszeichenfolge.

Schritt 5: Den Code ausführen

Das folgende Codebeispiel erstellt eine Amazon DocumentDB-Verbindung und druckt alle Datenbanken:

```
public static void TestConnection() {
    try (MongoClient mongoClient = makeDbConnection(DATABASE_NAME, DB_USER_NAME,
 DB_PASSWORD, DB_CLUSTER_ENDPOINT, KEYSTORE_PASSWORD)) {
        List < String > databases = mongoClient.listDatabaseNames().into(new ArrayList
 < > ());
```

```
        System.out.println("Databases: " + databases);
    } catch (MongoException e) {
        System.err.println("MongoDB error: " + e.getMessage());
        throw new RuntimeException(e);
    }
}
```

Bewährte Methoden für Verbindungen

Die folgenden bewährten Methoden sollten Sie berücksichtigen, wenn Sie mit einem MongoDB-Java-Treiber eine Verbindung zu Amazon DocumentDB herstellen:

- Schließen Sie Ihren immer `MongoClient`, wenn Sie den Client nicht mehr benötigen, um Ressourcen freizugeben.
- Behandeln Sie Ausnahmen angemessen und implementieren Sie eine korrekte Fehlerprotokollierung.
- Verwenden Sie Umgebungsvariablen oder speichern AWS Secrets Manager Sie vertrauliche Informationen wie Benutzernamen und Passwörter.

Durchführung von CRUD-Operationen in Amazon DocumentDB mit Java

In diesem Abschnitt wird die Durchführung von CRUD-Vorgängen (Create, Read, Update, Delete) in Amazon DocumentDB mithilfe von MongoDB-Java-Treibern beschrieben.

Themen

- Dokumente erstellen und in eine DocumentDB-Sammlung einfügen
- Lesen und Abrufen von Daten aus einer DocumentDB-Sammlung
- Aktualisierung vorhandener Dokumente in einer DocumentDB-Sammlung
- Dokumente aus einer DocumentDB-Sammlung entfernen
- Fehlerbehandlung mit Wiederholungslogik

Dokumente erstellen und in eine DocumentDB-Sammlung einfügen

Durch das Einfügen von Dokumenten in Amazon DocumentDB können Sie Ihren Sammlungen neue Daten hinzufügen. Je nach Ihren Bedürfnissen und dem Datenvolumen, mit dem Sie arbeiten, gibt es mehrere Möglichkeiten, Einfügungen durchzuführen. Die grundlegendste Methode zum

Einfügen eines einzelnen Dokuments in die Sammlung ist. `insertOne()` Um mehrere Dokumente gleichzeitig einzufügen, können Sie die `insertMany()` Methode verwenden, mit der Sie eine Reihe von Dokumenten in einem einzigen Vorgang hinzufügen können. Eine andere Methode zum Einfügen vieler Dokumente in eine DocumentDB-Sammlung ist. `bulkWrite()` In diesem Handbuch besprechen wir all diese Methoden zum Erstellen von Dokumenten in einer DocumentDB-Sammlung.

`insertOne()`

Lassen Sie uns zunächst untersuchen, wie ein einzelnes Dokument in eine Amazon DocumentDBB-Sammlung eingefügt wird. Das Einfügen eines einzelnen Dokuments erfolgt mithilfe der Methode. `insertOne()` Diese Methode verwendet ein Formular BsonDocument zum Einfügen und gibt ein `InsertOneResult` Objekt zurück, das verwendet werden kann, um die Objekt-ID des neu eingefügten Dokuments abzurufen. Der folgende Beispielcode zeigt das Einfügen eines Restaurantdokuments in die Sammlung:

```
Document article = new Document()
    .append("restaurantId", "REST-21G145")
    .append("name", "Future-proofed Intelligent Bronze Hat")
    .append("cuisine", "International")
    .append("rating", new Document()
        .append("average", 1.8)
        .append("totalReviews", 267))
    .append("features", Arrays.asList("Outdoor Seating", "Live Music"));

try {
    InsertOneResult result = collection.insertOne(article);
    System.out.println("Inserted document with the following id: " +
 result.getInsertedId());
} catch (MongoWriteException e) {
    // Handle duplicate key or other write errors
    System.err.println("Failed to insert document: " + e.getMessage());
    throw e;
} catch (MongoException e) {
    // Handle other MongoDB errors
    System.err.println("MongoDB error: " + e.getMessage());
    throw e;
}
```

Achten Sie bei der Verwendung darauf`insertOne()`, eine angemessene Fehlerbehandlung einzubeziehen. Im obigen Code hat beispielsweise „`restaurantId`“ einen eindeutigen Index. Wenn Sie diesen Code erneut ausführen, wird daher Folgendes ausgelöst`MongoWriteException`:

```
Failed to insert document: Write operation error on server
 docdbCluster.docdb.amazonaws.com:27017.
Write error: WriteError{code=11000, message='E11000 duplicate key error collection:
 Restaurants index: restaurantId_1', details={}}.
```

Fügt viele ein ()

Die wichtigsten Methoden zum Einfügen vieler Dokumente in eine Sammlung sind insertMany () und. `bulkWrite()`

Die `insertMany()` Methode ist die einfachste Methode, um mehrere Dokumente in einem einzigen Vorgang einzufügen. Sie akzeptiert eine Liste von Dokumenten und fügt sie in die Sammlung ein. Diese Methode ist ideal, wenn Sie einen Stapel neuer Dokumente einfügen, die unabhängig voneinander sind und keine speziellen Verarbeitungen oder gemischten Operationen erfordern. Der folgende Code zeigt, wie JSON-Dokumente aus einer Datei gelesen und in die Sammlung eingefügt werden. Die `insertMany()` Funktion gibt ein `InsertManyResult` `InsertManyResult` Objekt zurück, mit dem die IDs aller eingefügten Dokumente abgerufen werden können.

```
// Read JSON file content
String content = new String(Files.readAllBytes(Paths.get(jsonFileName)));
JSONArray jsonArray = new JSONArray(content);

// Convert JSON articles to Documents
List < Document > restaurants = new ArrayList < > ();
for (int i = 0; i < jsonArray.length(); i++) {
    JSONObject jsonObject = jsonArray.getJSONObject(i);
    Document doc = Document.parse(jsonObject.toString());
    restaurants.add(doc);
}
//insert documents in collection
InsertManyResult result = collection.insertMany(restaurants);

System.out.println("Count of inserted documents: " + result.getInsertedIds().size());
```

`bulkWrite()`

Die `bulkWrite()` Methode ermöglicht die Ausführung mehrerer Schreibvorgänge (Einfügen, Aktualisieren, Löschen) in einem einzigen Stapel. Sie können es verwenden`bulkWrite()`, wenn Sie verschiedene Arten von Vorgängen in einem einzigen Stapel ausführen müssen, z. B. wenn Sie einige Dokumente einfügen und andere aktualisieren müssen. `bulkWrite()`unterstützt zwei Arten von Batch-Schreibvorgängen: geordnet und ungeordnet:

- Geordnete Operationen — (Standard) Amazon DocumentDB verarbeitet die Schreibvorgänge sequentiell und stoppt beim ersten Fehler. Dies ist nützlich, wenn es auf die Reihenfolge der Operationen ankommt, z. B. wenn spätere Operationen von früheren Vorgängen abhängen. Geordnete Operationen sind jedoch im Allgemeinen langsamer als ungeordnete Operationen. Bei geordneten Vorgängen müssen Sie den Fall beheben, dass der Batch beim ersten Fehler stoppt und möglicherweise einige Operationen unbearbeitet bleiben.
- Ungeordnete Operationen — Ermöglicht Amazon DocumentDB, Inserts als einzelne Ausführung in der Datenbank zu verarbeiten. Tritt bei einem Dokument ein Fehler auf, wird der Vorgang mit den übrigen Dokumenten fortgesetzt. Dies ist besonders nützlich, wenn Sie große Datenmengen einfügen und einige Fehler tolerieren können, z. B. bei der Datenmigration oder bei Massenimporten, bei denen einige Dokumente aufgrund doppelter Schlüssel fehlschlagen könnten. Bei ungeordneten Vorgängen müssen Sie Szenarien mit teilweisem Erfolg berücksichtigen, in denen einige Operationen erfolgreich sind, während andere fehlschlagen.

Bei der Arbeit mit der `bulkWrite()` Methode sind einige wichtige Klassen erforderlich. Erstens dient die `WriteModel` Klasse als Basisklasse für alle Schreiboperationen und für spezifische Implementierungen wie `InsertOneModel`, `UpdateOneModel` `UpdateManyModel` `DeleteOneModel`, und für die `DeleteManyModel` Behandlung verschiedener Arten von Operationen.

Die `BulkWriteOptions` Klasse ist erforderlich, um das Verhalten von Massenoperationen zu konfigurieren, z. B. um die ordered/unordered Ausführung festzulegen oder die Dokumentvalidierung zu umgehen. Die `BulkWriteResult` Klasse bietet detaillierte Informationen zu den Ausführungsergebnissen, einschließlich der Anzahl der eingefügten, aktualisierten und gelöschten Dokumente.

Für die Fehlerbehandlung ist die `MongoBulkWriteException` Klasse von entscheidender Bedeutung, da sie Informationen über alle Fehler während des Massenvorgangs enthält, während die `BulkWriteError` Klasse spezifische Details zu einzelnen Betriebsfehlern enthält. Der folgende Code zeigt ein Beispiel für das Einfügen einer Liste von Dokumenten sowie das Aktualisieren und Löschen eines einzelnen Dokuments, und das alles im Rahmen eines einzelnen `bulkWrite()` Methodenaufrufs. Der Code zeigt auch, wie mit `BulkWriteOptions` und gearbeitet wird `BulkWriteResult`, sowie die korrekte Fehlerbehandlung der `bulkWrite()` Operation.

```
List < WriteModel < Document >> bulkOperations = new ArrayList < > ();
// get list of 10 documents representing 10 restaurants
List < Document > restaurantsToInsert = getSampleData();
```

```
for (Document doc: restaurantsToInsert) {
    bulkOperations.add(new InsertOneModel < > (doc));
}
// Update operation
bulkOperations.add(new UpdateOneModel < > (
    new Document("restaurantId", "REST-Y2E9H5"),
    new Document("", new Document("stats.likes", 20))
    .append("", new Document("rating.average", 4.5))));
// Delete operation
bulkOperations.add(new DeleteOneModel < > (new Document("restaurantId", "REST-
D2L431")));

// Perform bulkWrite operation
try {
    BulkWriteOptions options = new BulkWriteOptions()
        .ordered(false); // Allow unordered inserts

    BulkWriteResult result = collection.bulkWrite(bulkOperations, options);

    System.out.println("Inserted: " + result.getInsertedCount());
    System.out.println("Updated: " + result.getModifiedCount());
    System.out.println("Deleted: " + result.getDeletedCount());
} catch (MongoBulkWriteException e) {
    System.err.println("Bulk write error occurred: " + e.getMessage());
    // Log individual write errors
    for (BulkWriteError error: e.getWriteErrors()) {
        System.err.printf("Error at index %d: %s (Code: %d)%n", error.getIndex(),
 error.getMessage(),
            error.getCode());

        // Log the problematic document
        Document errorDoc = new Document(error.getDetails());
        if (errorDoc != null) {
            System.err.println("Problematic document: " + errorDoc);
        }
    }
} catch (Exception e) {
    System.err.println("Error during bulkWrite: " + e.getMessage());
}
```

Wiederholbare Schreibvorgänge

Im Gegensatz zu MongoDB unterstützt Amazon DocumentDB keine wiederholbaren Schreibversuche. Daher müssen Sie in ihren Anwendungen eine benutzerdefinierte Wiederholungslogik implementieren, insbesondere zur Behandlung von Netzwerkproblemen oder vorübergehenden Dienstausfällen. Eine gut implementierte Wiederholungsstrategie besteht in der Regel darin, die Verzögerung zwischen Wiederholungsversuchen zu erhöhen und die Gesamtzahl der Wiederholungen zu begrenzen. Fehlerbehandlung mit WiederholungslogikIm Folgenden finden Sie ein Codebeispiel für die Erstellung einer Wiederholungslogik mit Fehlerbehandlung.

Lesen und Abrufen von Daten aus einer DocumentDB-Sammlung

Das Abfragen von Dokumenten in Amazon DocumentDB basiert auf mehreren Schlüsselkomponenten, mit denen Sie Daten präzise abrufen und bearbeiten können. Die `find()` Methode ist die grundlegende Abfrage-API in MongoDB-Java-Treibern. Sie ermöglicht das Abrufen komplexer Daten mit zahlreichen Optionen zum Filtern, Sortieren und Projizieren von Ergebnissen. Neben der `find()` Methode `FindIterable` gibt es zwei weitere grundlegende Komponenten, `Filters` die die Bausteine für Abfrageoperationen in MongoDB-Java-Treibern bereitstellen.

Die `Filters` Klasse ist eine Hilfsklasse im MongoDB-Java-Treiber, die eine flüssige API zum Erstellen von Abfragefiltern bereitstellt. Diese Klasse bietet statische Factory-Methoden, die Instanzen von `Bson` Objekten erstellen, die verschiedene Abfragebedingungen repräsentieren. Zu den am häufigsten verwendeten Methoden gehören `eq()` Gleichheitsvergleiche,`gt()`, `lt()gte()`, und `lte()` für numerische Vergleiche `and()` sowie `or()` für das Kombinieren mehrerer Bedingungen `in()` sowie `nin()` für Tests der Array-Zugehörigkeit und `regex()` für den Mustervergleich. Die Klasse ist typsicher konzipiert und bietet im Vergleich zu dokumentbasierten Rohabfragen eine bessere Überprüfung bei der Kompilierung. Daher ist sie der bevorzugte Ansatz für die Erstellung von DocumentDB-Abfragen in Java-Anwendungen. Die Fehlerbehandlung ist robust, und es werden klare Ausnahmen für ungültige Filterkonstruktionen ausgelöst.

`FindIterable`ist eine spezielle Schnittstelle, die entwickelt wurde, um das Ergebnis der `find()` Methode zu verarbeiten. Es bietet eine Vielzahl von Methoden zur Verfeinerung und Steuerung der Abfrageausführung und bietet eine flüssige API für die Methodenverkettung. Die Schnittstelle umfasst wichtige Methoden zur Änderung von Abfragen, z. B. `limit()` zur Beschränkung der Anzahl der zurückgegebenen Dokumente, `skip()` zur Seitennummerierung, zur Sortierung von Ergebnissen, `sort()` zur Auswahl bestimmter Felder und `projection()` zur Indexauswahl. `hint()` Die Batch-, Skip- und Limit-Operationen in `FindIterable` sind wichtige Paginierungs- und Datenmanagement-Tools, mit denen Sie kontrollieren können, wie Dokumente aus der Datenbank abgerufen und verarbeitet werden.

Batching (`batchSize`) steuert, wie viele Dokumente DocumentDB in einem einzigen Netzwerk-Roundtrip an den Client zurückgibt. Wenn Sie eine Stapelgröße festlegen, gibt DocumentDB nicht alle passenden Dokumente auf einmal zurück, sondern gibt sie stattdessen in Gruppen der angegebenen Stapelgröße zurück.

Mit Skip können Sie den Startpunkt Ihrer Ergebnisse verschieben und DocumentDB im Wesentlichen anweisen, eine bestimmte Anzahl von Dokumenten zu überspringen, bevor Treffer zurückgegeben werden. Beispielsweise `skip(20)` werden die ersten 20 passenden Dokumente umgangen. Dies wird häufig in Paginierungsszenarien verwendet, in denen Sie nachfolgende Ergebnisseiten abrufen möchten.

Limit schränkt die Gesamtzahl der Dokumente ein, die von einer Abfrage zurückgegeben werden können. Wenn Sie angeben`limit(n)`, beendet DocumentDB die Rückgabe von Dokumenten, nachdem es 'n' Dokumente zurückgegeben hat, auch wenn die Datenbank mehr Treffer enthält.

`FindIterable`unterstützt sowohl Iterator- als auch Cursormuster beim Abrufen von Dokumenten aus Amazon DocumentDB. Der Vorteil der Verwendung `FindIterable` als Iterator besteht darin, dass Dokumente verzögert geladen werden können und Dokumente nur abgerufen werden, wenn sie von der Anwendung angefordert werden. Ein weiterer Vorteil der Verwendung des Iterators besteht darin, dass Sie nicht für die Aufrechterhaltung der Verbindung zum Cluster verantwortlich sind und daher kein ausdrückliches Schließen der Verbindung erforderlich ist.

`FindIterable`bieten auch Unterstützung `MongoCursor`, die die Verwendung von Cursormustern bei der Arbeit mit Amazon DocumentDB-Abfragen ermöglicht. `MongoCursor`ist eine für den MongoDB-Java-Treiber spezifische Implementierung, die die Kontrolle über Datenbankoperationen und Ressourcenmanagement bietet. Sie implementiert die `AutoCloseable` Schnittstelle und ermöglicht eine explizite Ressourcenverwaltung durch Try-with-Resources-Blöcke. Dies ist entscheidend für das ordnungsgemäße Schließen von Datenbankverbindungen und das Freigeben von Serverressourcen. Standardmäßig läuft das Timeout des Cursors nach 10 Minuten ab, und DocumentDB bietet Ihnen nicht die Möglichkeit, dieses Timeout-Verhalten zu ändern. Wenn Sie mit Stapeldaten arbeiten, stellen Sie sicher, dass Sie den nächsten Datenstapel abrufen, bevor das Zeitlimit für den Cursor überschritten wird. Eine wichtige Überlegung bei der Verwendung `MongoCursor` ist, dass ein explizites Schließen erforderlich ist, um Ressourcenlecks zu verhindern.

In diesem Abschnitt werden mehrere Beispiele für `find()` `Filters` und `FindIterable` vorgestellt.

Das folgende Codebeispiel zeigt, wie ein einzelnes Dokument mithilfe des Felds „RestaurantID“ abgerufen wird: `find()`

```
Document filter = new Document("restaurantId", "REST-21G145");
Document result = collection.find(filter).first();
```

Obwohl die Verwendung `Filters` eine bessere Überprüfung von Fehlern bei der Kompilierung ermöglicht, können Sie mit dem Java-Treiber auch direkt in der `find()` Methode einen `Bson` Filter angeben. Der folgende Beispielcode übergibt `Bson` das Dokument an`find()`:

```
result = collection.find(new Document("$and", Arrays.asList(
    new Document("rating.totalReviews", new Document("$gt", 1000)),
    new Document("priceRange", "$$"))))
```

Der nächste Beispielcode zeigt mehrere Beispiele für die Verwendung der `Filters` Klasse mit`find()`:

```
FindIterable < Document > results;

// Exact match
results = collection.find(Filters.eq("name", "Thai Curry Palace"));

// Not equal
results = collection.find(Filters.ne("cuisine", "Thai"));

// find an element in an array
results = collection.find(Filters.in("features", Arrays.asList("Private Dining")));

// Greater than
results = collection.find(Filters.gt("rating.average", 3.5));

// Between (inclusive)
results = collection.find(Filters.and(
    Filters.gte("rating.totalReviews", 100),
    Filters.lte("rating.totalReviews", 200)));
// AND
results = collection.find(Filters.and(
    Filters.eq("cuisine", "Thai"),
    Filters.gt("rating.average", 4.5)));

// OR
results = collection.find(Filters.or(
    Filters.eq("cuisine", "Thai"),
    Filters.eq("cuisine", "American")));
```

```
// All document where the Field exists
results = collection.find(Filters.exists("michelin"));

// Regex
results = collection.find(Filters.regex("name", Pattern.compile("Curry",
 Pattern.CASE_INSENSITIVE)));

// Find all document where the array contain the list of value regardless of its order
results = collection.find(Filters.all("features", Arrays.asList("Private Dining",
 "Parking")));

// Array size
results = collection.find(Filters.size("features", 4));
```

Das folgende Beispiel zeigt, wie die Operationen von`sort()`, `skip()``limit()`, und für ein `FindIterable` Objekt `batchSize()` verkettet werden. Die Reihenfolge, in der diese Operationen bereitgestellt werden, wirkt sich auf die Leistung Ihrer Abfrage aus. Als bewährte Methode sollte die Reihenfolge dieser Operationen`sort()`, `projection()``skip()`, `limit()` und lauten`batchSize()`.

```
FindIterable < Document > results = collection.find(Filters.gt("rating.totalReviews",
 1000))
    // Sorting
    .sort(Sorts.orderBy(
        Sorts.descending("address.city"),
        Sorts.ascending("cuisine")))
    // Field selection
    .projection(Projections.fields(
        Projections.include("name", "cuisine", "priceRange"),
        Projections.excludeId()))

    // Pagination
    .skip(20)
    .limit(10)
    .batchSize(2);
```

Der folgende Beispielcode zeigt das Erstellen eines Iterators auf`FindIterable`. Es verwendet das `forEach` Java-Konstrukt, um die Ergebnismenge zu durchlaufen.

```
collection.find(Filters.eq("cuisine", "American")).forEach(doc ->
 System.out.println(doc.toJson()));
```

Im letzten `find()` Codebeispiel wird gezeigt, wie es `cursor()` für das Abrufen von Dokumenten verwendet wird. Es erstellt den Cursor im Try-Block, wodurch sichergestellt wird, dass der Cursor geschlossen wird, wenn der Code den Try-Block verlässt.

```
try (MongoCursor < Document > cursor = collection.find(Filters.eq("cuisine",
 "American"))
    .batchSize(25)
    .cursor()) {
    while (cursor.hasNext()) {
        Document doc = cursor.next();
        System.out.println(doc.toJson());
    }
} // Cursor automatically closed
```

Aktualisierung vorhandener Dokumente in einer DocumentDB-Sammlung

Amazon DocumentDB bietet flexible und leistungsstarke Mechanismen zum Ändern vorhandener Dokumente und zum Einfügen neuer Dokumente, wenn sie nicht existieren. Der MongoDB-Java-Treiber bietet mehrere Methoden für Aktualisierungen: `updateOne()` für einzelne Dokumentaktualisierungen, `updateMany()` für mehrere Dokumentaktualisierungen und `replaceOne()` für den vollständigen Austausch von Dokumenten. Neben diesen drei Methoden `UpdateResult` gibt es noch weitere grundlegende Komponenten `Updates` `UpdateOptions`, die die Bausteine für Aktualisierungsvorgänge in MongoDB-Java-Treibern bilden.

Die `Updates` Klasse im MongoDB-Java-Treiber ist eine Hilfsklasse, die statische Factory-Methoden zum Erstellen von Aktualisierungsoperatoren bereitstellt. Sie dient als primärer Builder für die typsichere und lesbare Konstruktion von Aktualisierungsoperationen. Grundlegende Methoden wie `set()unset()`, und `inc()` ermöglichen die direkte Änderung der Dokumente. Die Stärke dieser Klasse wird deutlich, wenn mehrere Operationen mithilfe der `Updates.combine()` Methode kombiniert werden, mit der mehrere Aktualisierungsvorgänge atomar ausgeführt werden können, wodurch die Datenkonsistenz gewährleistet wird.

`UpdateOptions`ist eine leistungsstarke Konfigurationsklasse im Java-Treiber von MongoDB, die wichtige Anpassungsfunktionen für Dokumentaktualisierungsvorgänge bietet. Zwei wichtige Aspekte dieser Klasse sind die Unterstützung von Upsert- und Array-Filtern für Aktualisierungsvorgänge. Die Upsert-Funktion, die durch aktiviert wird`upsert(true)`, ermöglicht die Erstellung neuer Dokumente,

wenn bei einem Aktualisierungsvorgang keine passenden Dokumente gefunden werden. Durch `arrayFilters()` den Aktualisierungsvorgang können Array-Elemente, die bestimmte Kriterien erfüllen, präzise aktualisiert werden.

`UpdateResult`Der Java-Treiber von MongoDB bietet den Feedback-Mechanismus, der das Ergebnis eines Aktualisierungsvorgangs detailliert beschreibt. Diese Klasse umfasst drei wichtige Kennzahlen: die Anzahl der Dokumente, die den Aktualisierungskriterien entsprechen (`matchedCount`), die Anzahl der tatsächlich geänderten Dokumente (`modifiedCount`) und Informationen über alle fehlerhaften Dokumente (). `upsertedId` Das Verständnis dieser Metriken ist für eine korrekte Fehlerbehandlung, die Überprüfung von Aktualisierungsvorgängen und die Aufrechterhaltung der Datenkonsistenz in Anwendungen unerlässlich.

Ein einzelnes Dokument aktualisieren und ersetzen

In DocumentDB kann das Aktualisieren eines einzelnen Dokuments mit der updateOne () -Methode durchgeführt werden. Diese Methode verwendet einen Filterparameter, der normalerweise von der `Filters` Klasse bereitgestellt wird, um das zu aktualisierende Dokument zu identifizieren, einen `Updat` e-Parameter, der bestimmt, welche Felder aktualisiert werden sollen, und einen optionalen `UpdateOptions` Parameter, um verschiedene Optionen für die Aktualisierung festzulegen. Bei Verwendung der `updateOne()` Methode wird nur das erste Dokument aktualisiert, das den Auswahlkriterien entspricht. Der folgende Beispielcode aktualisiert ein einzelnes Feld eines Dokuments:

```
collection.updateOne(Filters.eq("restaurantId", "REST-Y2E9H5"),
    Updates.set("name", "Amazing Japanese sushi"));
```

Um mehrere Felder in einem Dokument zu aktualisieren, verwenden Sie `updateOne()` with, `Update.combine()` wie im folgenden Beispiel gezeigt. Dieses Beispiel zeigt auch, wie ein Element zu einem Array im Dokument hinzugefügt wird.

```
List<Bson> updates = new ArrayList<>();
// Basic field updates
updates.add(Updates.set("name", "Shanghai Best"));
// Array operations
updates.add(Updates.addEachToSet("features", Arrays.asList("Live Music")));
// Counter updates
updates.add(Updates.inc("rating.totalReviews", 10));
// Combine all updates
Bson combinedUpdates = Updates.combine(updates);
```

```
// Execute automic update with one call
collection.updateOne(Filters.eq("restaurantId","REST-1J83NH"), combinedUpdates);
```

Das folgende Codebeispiel zeigt, wie ein Dokument in der Datenbank aktualisiert wird. Wenn das angegebene Dokument nicht existiert, fügt der Vorgang es stattdessen automatisch als neues Dokument ein. Dieser Code zeigt auch, wie die über das `UpdateResult` Objekt verfügbaren Metriken verwendet werden.

```
Bson filter = Filters.eq("restaurantId", "REST-0Y9GL0");
Bson update = Updates.set("cuisine", "Indian");
// Upsert operation
UpdateOptions options = new UpdateOptions().upsert(true);
UpdateResult result = collection.updateOne(filter, update, options);

if (result.getUpsertedId() != null) {
    System.out.println("Inserted document with _id: " + result.getUpsertedId());
} else {
     System.out.println("Updated " + result.getModifiedCount() + " document(s)");
}
```

Das folgende Codebeispiel zeigt, wie ein vorhandenes Dokument mithilfe der `replaceOne()` Methode vollständig durch ein neues Dokument ersetzt wird, anstatt einzelne Felder zu aktualisieren. Die `replaceOne()` Methode überschreibt das gesamte Dokument und behält nur das `_id` Feld des Originals bei. Wenn mehrere Dokumente den Filterkriterien entsprechen, wird nur das zuerst gefundene Dokument ersetzt.

```
Document newDocument = new Document()
                .append("restaurantId", "REST-0Y9GL0")
                .append("name", "Bhiryani Adda")
                .append("cuisine", "Indian")
                .append("rating", new Document()
                        .append("average", 4.8)
                        .append("totalReviews", 267))
                .append("features", Arrays.asList("Outdoor Seating", "Live Music"));

UpdateResult result = collection.replaceOne(
                    Filters.eq("restaurantId", "REST-0Y9GL0"),
                    newDocument);
System.out.printf("Modified %d document%n", result.getModifiedCount());
```

Aktualisiere mehrere Dokumente

Es gibt zwei Möglichkeiten, mehrere Dokumente in einer Sammlung gleichzeitig zu aktualisieren. Sie können die `updateMany()` Methode verwenden oder die `UpdateManyModel` mit der `bulkWrite()` Methode verwenden. Die `updateMany()` Methode verwendet einen Filterparameter, um Dokumente für die Aktualisierung auszuwählen, den `Update` Parameter, um die zu aktualisierenden Felder zu identifizieren, und einen optionalen `UpdateOptions` Parameter, um Aktualisierungsoptionen anzugeben.

Der folgende Beispielcode veranschaulicht die Verwendung der `updateMany()` Methode:

```
Bson filter = Filters.and(
    Filters.in("features", Arrays.asList("Private Dining")),
    Filters.eq("cuisine", "Thai"));
UpdateResult result1 = collection.updateMany(filter, Updates.set("priceRange", "$$$"));
```

Der folgende Beispielcode demonstriert die `bulkWrite()` Methode mit demselben Update:

```
BulkWriteOptions options = new BulkWriteOptions().ordered(false);
List < WriteModel < Document >> updates = new ArrayList < > ();
Bson filter = Filters.and(
    Filters.in("features", Arrays.asList("Private Dining")),
    Filters.eq("cuisine", "Indian"));
Bson updateField = Updates.set("priceRange", "$$$");
updates.add(new UpdateManyModel < > (filter, updateField));
BulkWriteResult result = collection.bulkWrite(updates, options);
System.out.printf("Modified %d document%n", result.getModifiedCount());
```

Dokumente aus einer DocumentDB-Sammlung entfernen

Der MongoDB-Java-Treiber bietet `deleteOne()` die Möglichkeit, ein einzelnes Dokument zu entfernen und `deleteMany()` mehrere Dokumente zu entfernen, die bestimmten Kriterien entsprechen. Genau wie beim Aktualisieren kann auch der Löschvorgang mit der `bulkWrite()` Methode verwendet werden. Beide `deleteOne()` und `deleteMany()` geben ein `DeleteResult` Objekt zurück, das Informationen über das Ergebnis des Vorgangs liefert, einschließlich der Anzahl der gelöschten Dokumente. Das Folgende ist ein Beispiel für die Verwendung `deleteMany()` zum Entfernen mehrerer Dokumente:

```
Bson filter = Filters.and(
    Filters.eq("cuisine", "Thai"),
```

```
    Filters.lt("rating.totalReviews", 50));
DeleteResult result = collection.deleteMany(filter);
System.out.printf("Deleted %d document%n", result.getDeletedCount());
```

Fehlerbehandlung mit Wiederholungslogik

Eine robuste Fehlerbehandlungsstrategie für Amazon DocumentDB sollte die Kategorisierung von Fehlern in wiederholbare Fehler (wie Netzwerk-Timeouts, Verbindungsprobleme) und Fehler, die nicht wiederholt werden können (wie Authentifizierungsfehler, ungültige Abfragen), beinhalten. Bei Betriebsausfällen aufgrund von Fehlern, die erneut versucht werden sollten, sollte eine Zeitverzögerung zwischen den einzelnen Wiederholungsversuchen sowie die maximale Anzahl der Wiederholungsversuche eingeführt werden. Die CRUD-Operationen sollten sich in einem Try-Catch-Block befinden, der Catches und seine Unterklassen abfängt. `MongoException` Darüber hinaus sollte es die Überwachung und Protokollierung von Fehlern beinhalten, um die betriebliche Sichtbarkeit zu gewährleisten. Im Folgenden finden Sie einen Beispielcode, der zeigt, wie die Behandlung von Wiederholungsfehlern implementiert wird:

```
int MAX_RETRIES = 3;
int INITIAL_DELAY_MS = 1000;
int retryCount = 0;

while (true) {
    try {
        crud_operation(); //perform crud that will throw MongoException or one of its
 subclass
        break;
    } catch (MongoException e) {
        if (retryCount < MAX_RETRIES) {
            retryCount++;
            long delayMs = INITIAL_DELAY_MS * (long) Math.pow(2, retryCount - 1);
            try {
                TimeUnit.MILLISECONDS.sleep(delayMs);
            } catch (InterruptedException t) {
                Thread.currentThread().interrupt();
                throw new RuntimeException("Retry interrupted", t);
            }
            continue;
        } else
            throw new RuntimeException("Crud operation failed", e);
    }
}
```

Indexverwaltung in Amazon DocumentDB mit Java

Indizes ermöglichen das effiziente Abrufen von Daten aus einer Amazon DocumentDB-Sammlung. Ohne Indizes muss DocumentDB jedes Dokument in der Sammlung scannen, um Ergebnisse zurückzugeben, die einer bestimmten Abfrage entsprechen. Dieses Thema enthält Informationen zum Erstellen, Löschen und Auflisten von Indizes mithilfe der MongoDB-Java-Treiber. Außerdem wird beschrieben, wie festgestellt werden kann, ob ein bestimmter Index in der Abfrage verwendet wird, und wie Amazon DocumentDB Hinweise zur Verwendung eines bestimmten Indexes gegeben werden kann.

Themen

- Indizes mit Java erstellen
- Löschen von Indizes
- Festlegung der Indexauswahl und Bereitstellung eines Indexhinweises

Amazon DocumentDB unterstützt viele Arten von Indizes. Einen umfassenden Überblick über alle unterstützten Indizes finden Sie im Datenbank-Blog unter So indexieren Sie auf Amazon DocumentDB. AWS

Indizes mit Java erstellen

Es gibt zwei Mechanismen zum Erstellen von Indizes in Amazon DocumentDB mithilfe von MongoDB-Java-Treibern: durch `runCommand()` und entweder durch die `createIndex()` Methode für einen einzelnen Index oder die `createIndexes()` Methode für mehrere Indizes. Ein Grund für die Verwendung der `createIndexes()` Methoden `createIndex()` und ist, dass Sie eine bessere Fehlerbehandlung entwickeln können, indem Sie bestimmte Fehler im Zusammenhang mit der Indexerstellung abfangen. Ein weiterer Grund für die Verwendung dieser Methoden `runCommand()` ist, dass der MongDB-Java-Treiber eine Vielzahl unterstützender Klassen für die Indexerstellung und -bearbeitung bietet. Beachten Sie, dass diese unterstützenden Klassen nur verwendet werden können, wenn Sie die `createIndex()` `createIndexes()` OR-Methoden verwenden. Es gibt drei unterstützende Klassen:

- **`Indexes`**— Diese Klasse dient als Hilfsklasse und bietet statische Factory-Methoden zum Erstellen verschiedener Arten von Indizes. Sie vereinfacht das Erstellen komplexer Indexdefinitionen und wird häufig in Verbindung mit anderen indexbezogenen Klassen verwendet.
- **`IndexModel`**— Dies ist eine grundlegende Klasse, die sowohl die Definition der Indexschlüssel als auch ihre Optionen kapselt. Sie stellt eine vollständige Indexspezifikation dar, die kombiniert, was

indexiert werden soll (die Schlüssel), mit der Art der Indizierung (die Optionen). Diese Klasse ist besonders nützlich, wenn Sie mehrere Indizes gleichzeitig erstellen, da Sie damit eine Sammlung von Indexspezifikationen definieren können, die an die `createIndexes()` Methode übergeben werden können.

- **`IndexOptions`**— Dies ist eine umfassende Konfigurationsklasse, die eine Vielzahl von Methoden zur Anpassung des Indexverhaltens bietet. Sie umfasst Einstellungen für eindeutige Indizes, Indizes mit geringer Dichte, die Ablaufzeit (TTL) und partielle Filterausdrücke. Durch Methodenverkettung können Sie mehrere Optionen wie die Indexerstellung im Hintergrund und eindeutige Einschränkungen konfigurieren.

Erstellen Sie einen einzelnen Index

Dieses Beispiel zeigt, wie Sie mit der`createIndex()` -Methode im Hintergrund einen einzelnen Index erstellen. Informationen zur Erstellung von Hintergrund- und Vordergrundindizes finden Sie unterTypen zum Erstellen von Indizes. Im folgenden Codebeispiel wird `IndexOptions` ein eindeutiger Index mit dem Namen „unique_restaurantId_idx“ im Hintergrund erstellt. Dieses `IndexOptions` Objekt wird dann an die Methode übergeben. `createIndex()`

```
collection.createIndex(
    Indexes.ascending("restaurantId"),
    new IndexOptions()
        .unique(true)
        .name("unique_restaurantId_idx")
        .background(true));
```

Erstellen Sie mehrere Indizes

In diesem Beispiel werden mithilfe der Methode mehrere Indizes erstellt. `createIndexes()` Es erstellt zuerst die Option für jeden Index mithilfe des `IndexModel` Objekts und übergibt dann eine Liste von `IndexModel` Objekten an die `createIndexes()` Methode. Das folgende Codebeispiel zeigt, wie ein zusammengesetzter Index mithilfe der `Indexes` Dienstprogrammklasse erstellt wird. Diese Klasse wird auch verwendet, um anzugeben, ob Sie einen Index in aufsteigender oder absteigender Sortierreihenfolge erstellen möchten. Nach dem Erstellen mehrerer Indizes überprüft sie die Indexerstellung, indem sie die Methode aufruft. `listIndexes()`

```
// Single Field Index on cuisine
IndexModel singleIndex = new IndexModel(
    Indexes.ascending("cuisine"),
```

```
    new IndexOptions().name("cuisine_idx"));

// Compound Index
IndexModel compoundIndex = new IndexModel(
    Indexes.compoundIndex(
        Indexes.ascending("address.state"),
        Indexes.ascending("priceRange")),
    new IndexOptions().name("location_price_idx"));

// Build a list of IndexModel for the indexes
List < IndexModel > indexes = Arrays.asList(
    singleIndex,
    compoundIndex
);

collection.createIndexes(indexes);

// Verify created indexes
collection.listIndexes().forEach(index - > System.out.println("Created index: " +
 index.toJson()));
```

Erstellen Sie Indizes mit geringer Dichte und Teilindizes

Dieses Beispiel zeigt, wie ein Index mit geringer Dichte und ein Teilindex erstellt werden, indem `IndexModel` für jeden Indextyp ein Index erstellt wird.

```
// Sparse Index Model, this will identify only those documents that have a
// michelin star rating
IndexModel sparseIndex = new IndexModel(
    Indexes.ascending("michelin.star"),
    new IndexOptions()
    .name("michelin_sparse_idx")
    .sparse(true));

// Partial Index Model where the restaurant is active and has a rating of 4 and above
IndexModel partialIndex = new IndexModel(
    Indexes.ascending("rating.average"),
    new IndexOptions()
    .name("high_rated_active_idx")
    .partialFilterExpression(
        Filters.and(
            Filters.eq("isActive", true),
            Filters.gte("rating.average", 4.0))));
```

Erstellen Sie einen Textindex

Dieses Beispiel zeigt, wie ein Textindex erstellt wird. Für eine Sammlung ist nur ein Textindex zulässig, aber dieser eine Textindex kann ein zusammengesetzter Index sein, der mehrere Felder abdeckt. Wenn Sie mehrere Felder im Textindex verwenden, können Sie auch jedem der Felder im Index Gewichtungen zuweisen. Textindizes für Array-Felder werden von Amazon DocumentDB nicht unterstützt, und obwohl Sie bis zu 30 Felder im zusammengesetzten Textindex verwenden können, kann nur drei Feldern eine Gewichtung zugewiesen werden.

```
IndexModel textIndex = new IndexModel(
    new Document()
    .append("name", "text")
    .append("description", "text")
    .append("cuisine", "text"),
    new IndexOptions()
    .name("restaurant_text_idx")
    .weights(new Document()
        .append("name", 10) // Restaurant name gets highest weight
        .append("description", 5) // Description get medium weight
        .append("cuisine", 2) // Cuisine type gets low weight
    ));

collection.createIndex(textIndex.getKeys(), textIndex.getOptions());
```

Erstellen Sie einen Index mit **`runCommand()`**

Amazon DocumentDB unterstützt die parallele Indexerstellung, um die für die Erstellung von Indizes benötigte Zeit zu verkürzen. Bei der parallelen Indizierung werden mehrere Worker gleichzeitig verwendet. Für die Indexerstellung werden standardmäßig zwei Worker verwendet. Weitere Informationen finden Sie im AWS Datenbank-Blog unter Das Potenzial der parallelen Indizierung in Amazon DocumentDB nutzen. Derzeit unterstützen die MongDB-Java-Treiber die Angabe der Worker-Option nicht, wenn Sie `createIndex()` oder `createIndexes()` verwenden. Daher können Sie Worker nur über den angeben. `runCommand` Das folgende Codebeispiel zeigt, wie ein Index erstellt wird, der den Worker auf vier erhöht: `runCommand`

```
Document command = new Document("createIndexes", "Restaurants")
    .append("indexes", Arrays.asList(
        new Document("key", new Document("name", 1))
        .append("name", "restaurant_name_idx")
        .append("workers", 4) // Specify number of workers
    ));
```

```
Document commendResult = connectedDB.runCommand(command);
```

Löschen von Indizes

Der MongoDB-Java-Treiber bietet mehrere Methoden zum Löschen von Indizes, die auf verschiedene Szenarien und Ihre Präferenzen zugeschnitten sind. Sie können Indizes nach Namen, nach Schlüsselspezifikation oder alle Indizes auf einmal löschen. Die Methoden `dropIndex()` und `dropIndexes()` können für ein Auflistungsobjekt aufgerufen werden, um einen Index zu löschen. Wenn Sie einen Index anhand des Namens löschen, sollten Sie sicherstellen, dass der richtige Indexname verwendet wird, was möglicherweise nicht immer intuitiv ist, insbesondere bei zusammengesetzten oder automatisch generierten Indizes. Der Versuch, einen nicht existierenden Index zu löschen, führt zu einem. `MongoCommandException` Der `default _id` Index kann nicht gelöscht werden, da dadurch die Eindeutigkeit des Dokuments innerhalb der Sammlung gewährleistet wird.

Das folgende Codebeispiel zeigt, wie ein Index gelöscht wird, indem der Feldname angegeben wird, in dem der Index erstellt wird, oder indem alle Indizes gelöscht werden:

```
String indexName = "unique_restaurantId_idx";
Document keys = new Document("cuisine", 1);
// Drop index by name
collection.dropIndex(indexName);

// Drop index by keys
collection.dropIndex(keys);

// Drop all indexes
collection.dropIndexes();
```

Wenn Sie Indizes mit mehreren Schlüsseln löschen, stellen Sie sicher, dass ein zusammengesetzter Index vorhanden ist, der alle angegebenen Schlüssel enthält, und dass die Reihenfolge der Schlüssel korrekt ist. Der obige Beispielcode zur Indexerstellung zeigt einen zusammengesetzten Schlüssel zu „Küche“ und Funktionen. Wenn Sie versuchen, diesen zusammengesetzten Schlüssel zu löschen, aber die Reihenfolge nicht der Reihenfolge entspricht, die für die Erstellung verwendet wurde, tritt ein MongoCommnadException Fehler wie folgt auf:

```
Document keys = new Document("features", 1)
    .append("cuisine", 1);
try {
```

```
    // Drop index by keys
    collection.dropIndex(keys);
    System.out.println("Successfully dropped index with keys: " + keys.toJson());

} catch (MongoCommandException commErr) {
    System.out.println("Error dropping index: " + commErr.getErrorMessage());
    throw new RuntimeException("MongoCommandException was thrown while dropping index",
 commErr);
}
```

Der folgende Fehler wird angezeigt:

```
Error dropping index: Cannot drop index: index not found.
Tests run: 3, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 0.819 sec <<< FAILURE!
com.amazon.docdb.guide.DocDBGuideTest.testindexGuide()  Time elapsed: 0.817 sec  <<<
 FAILURE!
org.opentest4j.AssertionFailedError: Unexpected exception thrown:
 java.lang.RuntimeException: MongoCommandException was thrown while dropping index
```

Festlegung der Indexauswahl und Bereitstellung eines Indexhinweises

Die Arbeit mit der Erklärungsfunktion in Amazon DocumentDB ist entscheidend, um die Abfrageleistung und Indexnutzung zu verstehen. Wenn Sie eine Abfrage ausführen, können Sie die `explain()` Methode anhängen, um detaillierte Informationen über den Abfrageplan zu erhalten, einschließlich der Indizes, falls vorhanden, die verwendet werden. Die `explain()` Ausgabe bietet Einblicke in die Phasen der Abfrageausführung, die Anzahl der untersuchten Dokumente und die für jede Phase benötigte Zeit. Diese Informationen sind von unschätzbarem Wert, um festzustellen, ob ein bestimmter Index effektiv verwendet wird oder ob die Abfrage von einer anderen Indexstruktur profitieren könnte.

Die `explain()` Methode kann mit der Methode verkettet werden. `find()` Die `explain()` Methode kann eine optionale `ExplainVerbosity` Aufzählung annehmen, die den Grad der Ausführlichkeit bestimmt, der zurückgegeben wird. `explain()` Derzeit werden nur die `QUERY_PLANNER` Enumeratoren `EXECUTION_STATS` und von DocumentDB unterstützt. Das folgende Codebeispiel zeigt, wie Sie den Abfrageplaner für eine bestimmte Abfrage abrufen:

```
// Query we want to analyze
Document query = new Document()
    .append("cuisine", "Thai")
    .append("rating.average", new Document("$gte", 4.0));
```

```
Document allPlansExplain =
 collection.find(query).explain(ExplainVerbosity.QUERY_PLANNER);
System.out.println("All Plans Explain:\n" + allPlansExplain.toJson());
```

Das folgende JSON-Dokument wird für die Ausführlichkeitsstufe des Abfrageplaners zurückgegeben:

```
{
  "queryPlanner": {
    "plannerVersion": 1,
    "namespace": "ProgGuideData.Restaurants",
    "winningPlan": {
      "stage": "IXSCAN",
      "indexName": "cuisine_idx",
      "direction": "forward"
    }
  },
  "serverInfo": {
    "host": "guidecluster3",
    "port": 27017,
    "version": "5.0.0"
  },
  "ok": 1,
  "operationTime": {
    "$timestamp": {
      "t": 1739221668,
      "i": 1
    }
  }
}
```

Sie haben mehrere Möglichkeiten, Amazon DocumentDB zu beeinflussen oder zu zwingen, einen bestimmten Index zu verwenden. Mit den `hintString()` Methoden `hint()` und können Sie das Standardverhalten des Abfrageoptimierers bei der Indexauswahl außer Kraft setzen, indem Sie explizit angeben, welcher Index für eine Abfrage verwendet werden soll. Während der Abfrageoptimierer von DocumentDB im Allgemeinen eine gute Wahl für die Indexauswahl bietet, gibt es auch Szenarien, in denen das `hint()` Durchsetzen eines bestimmten Indexes von Vorteil sein `hintString()` kann, z. B. beim Umgang mit schiefen Daten oder beim Testen der Indexleistung.

Das folgende Codebeispiel erzwingt die Verwendung des zusammengesetzten Indexes „cuisine_features_idx“ für dieselbe Abfrage, die im obigen Code ausgeführt wurde:

```
// Query we want to analyze
Document query = new Document()
    .append("cuisine", "Thai")
    .append("rating.average", new Document("$gte", 4.0));

List < Document > queryDocs = new ArrayList < > ();
collection.find(query).hintString("cuisine_features_idx").forEach(doc - >
 queryDocs.add(doc));
```

Event-driven Programmierung mit Amazon DocumentDB und Java

Event-driven Die Programmierung im Kontext von Amazon DocumentDB stellt ein leistungsstarkes Architekturmuster dar, bei dem Datenbankänderungen als primäre Ereignisgeneratoren dienen, die nachfolgende Geschäftslogik und Prozesse auslösen. Wenn Datensätze in eine DocumentDB-Sammlung eingefügt, aktualisiert oder gelöscht werden, wirken diese Änderungen als Ereignisse, die automatisch verschiedene nachgelagerte Prozesse, Benachrichtigungen oder Datensynchronisierungsaufgaben auslösen. Dieses Muster ist besonders nützlich in modernen verteilten Systemen, in denen mehrere Anwendungen oder Dienste in Echtzeit auf Datenänderungen reagieren müssen. Der primäre Mechanismus zur Implementierung ereignisgesteuerter Programmierung in DocumentDB sind Change Streams.

> **Note**
>
> In diesem Handbuch wird davon ausgegangen, dass Sie Change Streams für eine Sammlung aktiviert haben, mit der Sie gerade arbeiten. Weitere Informationen Change Streams mit Amazon DocumentDB verwenden zum Aktivieren von Change-Streams für die Sammlung finden Sie unter.

Arbeiten mit Change-Streams aus der Java-Anwendung

Die `watch()` Methode im Java-Treiber von MongoDB ist der primäre Mechanismus zur Überwachung von Datenänderungen in Amazon DocumentDB in Echtzeit. Die `watch()` Methode kann von Objekten `MongoClient` `MongoDatabase`, und `MongoCollection` aufgerufen werden.

Die `watch()` Methode gibt eine Instanz zurück `ChangeStreamIterable`, die verschiedene Konfigurationsoptionen unterstützt, darunter die vollständige Dokumentsuche nach Aktualisierungen, die Bereitstellung von Lebenslauf-Tokens und Zeitstempeln für die Zuverlässigkeit sowie Pipeline-Aggregationsstufen zum Filtern von Änderungen.

ChangeStreamIterableimplementiert die Java-Kernschnittstelle `Iterable` und kann mit verwendet werden. `forEach()` Um Ereignisse mit zu erfassen`forEach()`, übergeben Sie eine Callback-Funktion`forEach()`, die das geänderte Ereignis verarbeitet. Der folgende Codeausschnitt zeigt, wie ein Change-Stream für eine Sammlung geöffnet wird, um die Überwachung von Änderungsereignissen zu starten:

```
ChangeStreamIterable < Document > iterator = collection.watch();
iterator.forEach(event - > {
    System.out.println("Received a change: " + event);
});
```

Eine andere Möglichkeit, alle Änderungsereignisse zu durchlaufen, besteht darin, einen Cursor zu öffnen, der eine Verbindung zum Cluster aufrechterhält und kontinuierlich neue Änderungsereignisse empfängt, sobald sie auftreten. Um einen Change-Streams-Cursor zu erhalten, verwenden Sie die `cursor()` Methode object. `ChangeStreamIterable` Das folgende Codebeispiel zeigt, wie Änderungsereignisse mithilfe des Cursors überwacht werden:

```
try (MongoChangeStreamCursor < ChangeStreamDocument < Document >> cursor =
 collection.watch().cursor()) {
    System.out.println(cursor.tryNext());
}
```

Es hat sich bewährt, entweder die Anweisung `MongoChangeStreamCursor` in a try-with-resource zu erstellen oder den Cursor manuell zu schließen. Wenn Sie die `cursor()` Methode aufrufen, wird eine `ChangeStreamIterable` zurückgegeben`MongoChangeStreamCursor`, die über einem Objekt erstellt wurde. `ChangeStreamDocument`

Die `ChangeStreamDocument` Klasse ist eine wichtige Komponente, die einzelne Änderungsereignisse im Stream darstellt. Sie enthält detaillierte Informationen zu jeder Änderung, einschließlich des Vorgangstyps (Einfügen, Aktualisieren, Löschen, Ersetzen), des Dokumentschlüssels, der Namespace-Informationen und des vollständigen Dokumentinhalts, sofern verfügbar. Die Klasse bietet Methoden für den Zugriff auf verschiedene Aspekte des Änderungsereignisses, z. B. `getOperationType()` um die Art der Änderung `getFullDocument()` zu bestimmen, auf den vollständigen Dokumentstatus zuzugreifen und das geänderte Dokument `getDocumentKey()` zu identifizieren.

Das `ChangeStreamDocument` Objekt stellt zwei wichtige Informationen bereit: ein Resume-Token und die Uhrzeit des Änderungsereignisses.

Resume-Token und zeitbasierte Operationen in DocumentDB-Change-Streams bieten entscheidende Mechanismen zur Aufrechterhaltung der Kontinuität und zur Verwaltung des Zugriffs auf historische Daten. Ein Resume-Token ist eine eindeutige Kennung, die für jedes Änderungsereignis generiert wird. Sie dient als Lesezeichen, das es Anwendungen ermöglicht, die Change-Stream-Verarbeitung nach Verbindungsabbrüchen oder Ausfällen von einem bestimmten Punkt aus neu zu starten. Wenn ein Change-Stream-Cursor erstellt wird, kann er über die `resumeAfter()` Option ein zuvor gespeichertes Resume-Token verwenden, sodass der Stream an der Stelle fortgesetzt werden kann, an der er aufgehört hat, anstatt am Anfang zu beginnen oder Ereignisse zu verlieren.

Time-based Operationen in Change-Streams bieten unterschiedliche Ansätze, um den Ausgangspunkt der Überwachung von Änderungsereignissen zu verwalten. Mit dieser `startAtOperationTime()` Option können Sie beginnen, Änderungen zu beobachten, die zu oder nach einem bestimmten Zeitstempel eingetreten sind. Diese zeitbasierten Funktionen sind besonders nützlich in Szenarien, in denen historische Daten verarbeitet, zu einem bestimmten Zeitpunkt wiederhergestellt oder zwischen Systemen synchronisiert werden müssen.

Im folgenden Codebeispiel wird das mit dem Einfügedokument verknüpfte Ereignis abgerufen, das zugehörige Resume-Token erfasst und dann dieses Token bereitgestellt, um mit der Überwachung von Ereignissen nach dem Insert-Ereignis zu beginnen. Das Ereignis wird dem Aktualisierungsereignis zugeordnet, ruft dann die Cluster-Uhrzeit ab, zu der die Aktualisierung stattgefunden hat, und verwendet diesen Zeitstempel als Ausgangspunkt für die weitere Verarbeitung.

```
BsonDocument resumeToken;
BsonTimestamp resumeTime;

try (MongoChangeStreamCursor < ChangeStreamDocument < Document >> cursor =
 collection.watch().cursor()) {
    System.out.println("****************** Insert Document ******************");
    ChangeStreamDocument < Document > insertChange = cursor.tryNext();
    resumeToken = insertChange.getResumeToken();
    printJson(cursor.tryNext());
}
try (MongoChangeStreamCursor < ChangeStreamDocument < Document >> cursor =
 collection.watch()
    .resumeAfter(resumeToken)
    .cursor()) {
    System.out.println("****************** Update Document ******************");
    ChangeStreamDocument < Document > insertChange = cursor.tryNext();
    resumeTime = insertChange.getClusterTime();
```

```
    printJson(cursor.tryNext());
}
try (MongoChangeStreamCursor < ChangeStreamDocument < Document >> cursor =
 collection.watch()
    .startAtOperationTime(resumeTime)
    .cursor()) {
    System.out.println("****************** Delete Document ******************");
    printJson(cursor.tryNext());
  }
```

Standardmäßig umfasst das Aktualisierungsereignis nicht das gesamte Dokument und nur die vorgenommenen Änderungen. Wenn Sie auf das komplette Dokument zugreifen müssen, das aktualisiert wurde, können Sie die `fullDocument()` Methode für das `ChangeStreamIterable` Objekt aufrufen. Denken Sie daran, dass, wenn Sie die Rückgabe eines vollständigen Dokuments für ein Aktualisierungsereignis anfordern, das Dokument zurückgegeben wird, das zum Zeitpunkt des Aufrufs von Change Streams vorhanden ist.

Diese Methode verwendet eine `FullDocument` Aufzählung als Parameter. Derzeit unterstützt Amazon DocumentDB nur DEFAULT und `UPDATE_LOOKUP` Werte. Der folgende Codeausschnitt zeigt, wie Sie das vollständige Dokument für Aktualisierungsereignisse anfordern, wenn Sie beginnen, auf Änderungen zu achten:

```
try (MongoChangeStreamCursor < ChangeStreamDocument < Document >> cursor =
 collection.watch().fullDocument(FullDocument.UPDATE_LOOKUP).cursor())
```

Verwenden der JSON-Schemavalidierung

Mithilfe des `$jsonSchema` Testabfrageoperators können Sie überprüfen, ob Dokumente in Ihre Sammlungen eingefügt werden.

Topics

- JSON-Schemavalidierung erstellen und verwenden
- Unterstützte Schlüsselwörter
- umgehen DocumentValidation
- Einschränkungen

JSON-Schemavalidierung erstellen und verwenden

Eine Sammlung mit Schemavalidierung erstellen

Sie können eine Sammlung mit `createCollection` Betriebs- und Validierungsregeln erstellen. Diese Validierungsregeln werden beim Einfügen oder Aktualisieren von Amazon DocumentDB DocumentDB-Dokumenten angewendet. Das folgende Codebeispiel zeigt Validierungsregeln für eine Sammlung von Mitarbeitern:

```
db.createCollection("employees", {
   "validator": {
      "$jsonSchema": {
         "bsonType": "object",
         "title": "employee validation",
         "required": [ "name", "employeeId"],
         "properties": {
            "name": {
                  "bsonType": "object",
                  "properties": {
                     "firstName": {
                        "bsonType": ["string"]
                     },
                     "lastName": {
                        "bsonType": ["string"]
                     }
                  },
                  "additionalProperties" : false
            },
            "employeeId": {
               "bsonType": "string",
               "description": "Unique Identifier for employee"
            },
             "salary": {
               "bsonType": "double"
            },
            "age": {
               "bsonType": "number"
            }
         },
         "additionalProperties" : true
      }
   },
   "validationLevel": "strict", "validationAction": "error"
```

```
} )
```

Ein gültiges Dokument einfügen

Im folgenden Beispiel werden Dokumente eingefügt, die den oben genannten Schemavalidierungsregeln entsprechen:

```
db.employees.insert({"name" : { "firstName" : "Carol" , "lastName" : "Smith"},
 "employeeId": "c720a" , "salary": 1000.0 })
db.employees.insert({ "name" : { "firstName" : "William", "lastName" : "Taylor" },
 "employeeId" : "c721a", "age" : 24})
```

Ein ungültiges Dokument wird eingefügt

Im folgenden Beispiel werden Dokumente eingefügt, die den obigen Schemavalidierungsregeln nicht entsprechen. In diesem Beispiel ist der EmployeeID-Wert keine Zeichenfolge:

```
db.employees.insert({
    "name" : { "firstName" : "Carol" , "lastName" : "Smith"},
    "employeeId": 720 ,
    "salary": 1000.0
})
```

Dieses Beispiel zeigt eine falsche Syntax innerhalb des Dokuments.

Eine Sammlung ändern

Der `collMod` Befehl wird verwendet, um Validierungsregeln einer vorhandenen Sammlung hinzuzufügen oder zu ändern. Im folgenden Beispiel wird der Liste der erforderlichen Felder ein Gehaltsfeld hinzugefügt:

```
db.runCommand({"collMod" : "employees",
   "validator": {
      "$jsonSchema": {
         "bsonType": "object",
         "title": "employee validation",
         "required": [ "name", "employeeId", "salary"],
         "properties": {
            "name": {
                  "bsonType": "object",
                  "properties": {
                     "firstName": {
```

```
                        "bsonType": ["string"]
                     },
                     "lastName": {
                        "bsonType": ["string"]
                     }
                  },
                  "additionalProperties" : false
            },
            "employeeId": {
               "bsonType": "string",
               "description": "Unique Identifier for employee"
            },
             "salary": {
               "bsonType": "double"
            },
            "age": {
               "bsonType": "number"
            }
         },
         "additionalProperties" : true
      }
   }
} )
```

Adressierung von Dokumenten, die vor der Änderung der Validierungsregeln hinzugefügt wurden

Verwenden Sie die folgenden `validationLevel` Modifikatoren, um Dokumente zu adressieren, die Ihrer Sammlung hinzugefügt wurden, bevor die Validierungsregeln geändert wurden:

- Strikt: Wendet Validierungsregeln auf alle Einfügungen und Aktualisierungen an.
- moderat: Wendet Validierungsregeln auf bestehende gültige Dokumente an. Bei Aktualisierungen werden vorhandene ungültige Dokumente nicht überprüft.

Im folgenden Beispiel ist nach der Aktualisierung der Validierungsregeln für die Sammlung mit dem Namen „Mitarbeiter“ das Feld Gehalt erforderlich. Die Aktualisierung des folgenden Dokuments schlägt fehl:

```
db.runCommand({
    update: "employees",
    updates: [{
        q: { "employeeId": "c721a" },
```

```
        u: { age: 25 , salary : 1000},
        upsert: true }]
})
```

Amazon DocumentDB gibt die folgende Ausgabe zurück:

```
{
"n" : 0,
    "nModified" : 0,
    "writeErrors" : [
        {
"index" : 0,
            "code" : 121,
            "errmsg" : "Document failed validation"
        }
    ],
    "ok" : 1,
    "operationTime" : Timestamp(1234567890, 1)
}
```

Wenn Sie die Validierungsstufe `moderate` auf aktualisieren, kann das obige Dokument erfolgreich aktualisiert werden:

```
db.runCommand({
    "collMod" : "employees",
    validationLevel : "moderate"
})

db.runCommand({
    update: "employees",
    updates: [{
        q: { "employeeId": "c721a" },
        u: { age: 25 , salary : 1000},
        upsert: true }]
})
```

Amazon DocumentDB gibt die folgende Ausgabe zurück:

```
{
"n" : 1,
    "nModified" : 1,
    "ok" : 1,
```

```
    "operationTime" : Timestamp(1234567890, 1)
}
```

Dokumente werden mit dem $JsonSchema abgerufen

Der `$jsonSchema` Operator kann als Filter verwendet werden, um Dokumente abzufragen, die dem JSON-Schema entsprechen. Dies ist ein Operator der obersten Ebene, der in Filterdokumenten als Feld der obersten Ebene vorhanden sein oder mit Abfrageoperatoren wie `$and` `$or`, und `$nor` verwendet werden kann. Die folgenden Beispiele zeigen die Verwendung von $jsonSchema als Einzelfilter und mit anderen Filteroperatoren:

Dokument, das in eine Sammlung „Mitarbeiter“ eingefügt wurde:

```
{ "name" : { "firstName" : "Carol", "lastName" : "Smith" }, "employeeId" : "c720a",
 "salary" : 1000 }
{ "name" : { "firstName" : "Emily", "lastName" : "Brown" }, "employeeId" : "c720b",
 "age" : 25, "salary" : 1050.2 }
{ "name" : { "firstName" : "William", "lastName" : "Taylor" }, "employeeId" : "c721a",
 "age" : 24, "salary" : 1400.5 }
{ "name" : { "firstName" : "Jane", "lastName" : "Doe" }, "employeeId" : "c721a",
 "salary" : 1300 }
```

Sammlung wurde nur mit dem `$jsonSchema` Operator gefiltert:

```
db.employees.find({
       $jsonSchema: { required: ["age"] } })
```

Amazon DocumentDB gibt die folgende Ausgabe zurück:

```
{ "_id" : ObjectId("64e5f91c6218c620cf0e8f8b"), "name" : { "firstName" : "Emily",
 "lastName" : "Brown" }, "employeeId" : "c720b", "age" : 25, "salary" : 1050.2 }
{ "_id" : ObjectId("64e5f94e6218c620cf0e8f8c"), "name" : { "firstName" : "William",
 "lastName" : "Taylor" }, "employeeId" : "c721a", "age" : 24, "salary" : 1400.5 }
```

Sammlung, die mit dem `$jsonSchema` Operator und einem anderen Operator gefiltert wurde:

```
db.employees.find({
       $or: [{ $jsonSchema: { required: ["age", "name"]}},
            { salary: { $lte:1000}}]});
```

Amazon DocumentDB gibt die folgende Ausgabe zurück:

```
{ "_id" : ObjectId("64e5f8886218c620cf0e8f8a"), "name" : { "firstName" : "Carol",
 "lastName" : "Smith" }, "employeeId" : "c720a", "salary" : 1000 }
{ "_id" : ObjectId("64e5f91c6218c620cf0e8f8b"), "name" : { "firstName" : "Emily",
 "lastName" : "Brown" }, "employeeId" : "c720b", "age" : 25, "salary" : 1050.2 }
{ "_id" : ObjectId("64e5f94e6218c620cf0e8f8c"), "name" : { "firstName" : "William",
 "lastName" : "Taylor" }, "employeeId" : "c721a", "age" : 24, "salary" : 1400.5 }
```

Sammlung, die mit dem `$jsonSchema` Operator und mit dem `$match` im Aggregatfilter enthaltenen Filter gefiltert wurde:

```
db.employees.aggregate(
    [{ $match: {
        $jsonSchema: {
            required: ["name", "employeeId"],
            properties: {"salary" :{"bsonType": "double"}}
        }
       }
    }]
)
```

Amazon DocumentDB gibt die folgende Ausgabe zurück:

```
{
"_id" : ObjectId("64e5f8886218c620cf0e8f8a"),
 "name" : { "firstName" : "Carol", "lastName" : "Smith" },
"employeeId" : "c720a",
"salary" : 1000
}
{
"_id" : ObjectId("64e5f91c6218c620cf0e8f8b"),
"name" : { "firstName" : "Emily", "lastName" : "Brown" },
"employeeId" : "c720b",
"age" : 25,
"salary" : 1050.2
}
{
"_id" : ObjectId("64e5f94e6218c620cf0e8f8c"),
"name" : { "firstName" : "William", "lastName" : "Taylor" },
"employeeId" : "c721a",
"age" : 24,
"salary" : 1400.5
}
```

```
{
"_id" : ObjectId("64e5f9786218c620cf0e8f8d"),
"name" : { "firstName" : "Jane", "lastName" : "Doe" },
"employeeId" : "c721a",
"salary" : 1300
}
```

Bestehende Validierungsregeln anzeigen

Um die vorhandenen Validierungsregeln für eine Sammlung zu sehen, verwenden Sie:

```
db.runCommand({
    listCollections: 1,
    filter: { name: 'employees' }
})
```

Amazon DocumentDB gibt die folgende Ausgabe zurück:

```
{
    "waitedMS" : NumberLong(0),
    "cursor" : {
        "firstBatch" : [
            {
                "name" : "employees",
                "type" : "collection",
                "options" : {
                    "autoIndexId" : true,
                    "capped" : false,
                    "validator" : {
                        "$jsonSchema" : {
                            "bsonType" : "object",
                            "title" : "employee validation",
                            "required" : [
                                "name",
                                "employeeId",
                                "salary"
                            ],
                            "properties" : {
                                "name" : {
                                    "bsonType" : "object",
                                    "properties" : {
                                        "firstName" : {
                                            "bsonType" : [
```

```
                                        "string"
                                    ]
                                },
                                "lastName" : {
                                    "bsonType" : [
                                        "string"
                                    ]
                                }
                            },
                            "additionalProperties" : false
                        },
                        "employeeId" : {
                            "bsonType" : "string",
                            "description" : "Unique Identifier for employee"
                        },
                        "salary" : {
                            "bsonType" : "double"
                        },
                        "age" : {
                            "bsonType" : "number"
                        }
                    },
                    "additionalProperties" : true
                }
            },
            "validationLevel" : "moderate",
            "validationAction" : "error"
        },
        "info" : {
            "readOnly" : false
        },
        "idIndex" : {
            "v" : 2,
            "key" : {
                "_id" : 1
            },
            "name" : "_id_",
            "ns" : "test.employees"
        }
    }
],
"id" : NumberLong(0),
"ns" : "test.$cmd.listCollections"
},
```

```
    "ok" : 1,
    "operationTime" : Timestamp(1692788937, 1)
}
```

Amazon DocumentDB behält auch die Validierungsregeln in der `$out` Aggregationsphase bei.

Unterstützte Schlüsselwörter

Die folgenden Felder werden in den `collMod` Befehlen `create` und unterstützt:

- **`Validator`**— Unterstützt den `$jsonSchem` Operator a.
- **`ValidationLevel`**— Unterstützt `moderate` Werte `off`strict, und.
- **`ValidationAction`**— Unterstützt den `error` Wert.

Der $jsonSchema-Operator unterstützt die folgenden Schlüsselwörter:

- `additionalItems`
- `additionalProperties`
- `allOf`
- `anyOf`
- `bsonType`
- `dependencies`
- `description`
- `enum`
- `exclusiveMaximum`
- `exclusiveMinimum`
- `items`
- `maximum`
- `minimum`
- `maxItems`
- `minItems`
- `maxLength`
- `minLength`
- `maxProperties`

- `minProperties`
- `multipleOf`
- `not`
- `oneOf`
- `pattern`
- `patternProperties`
- `properties`
- `required`
- `title`
- `type`
- `uniqueItems`

umgehen DocumentValidation

Amazon DocumentDB unterstützt by`passDocumentValidation` die folgenden Befehle und Methoden:

- `insert`
- `update`
- `findAndModify`
- $outStufe im `aggregate` Befehl und in der Methode `db.collection.aggregate()`

Amazon DocumentDB unterstützt die folgenden Befehle nicht für`bypassDocumentValidation`:

- `$merge`im `aggregate` Befehl und in der Methode `db.collection.aggregate()`
- `mapReduce`Befehl und `db.collection.mapReduce()` Methode
- `applyOps` command

Einschränkungen

Für die `$jsonSchema` Validierung gelten die folgenden Einschränkungen:

- Amazon DocumentDB gibt den Fehler „Dokument konnte nicht validiert werden“ zurück, wenn ein Vorgang die Validierungsregel nicht erfüllt.

- Elastische Amazon DocumentDB-Cluster werden nicht unterstützt`$jsonSchema`.

Abfragen in Amazon DocumentDB

In diesem Abschnitt werden alle Aspekte der Abfrage mit Amazon DocumentDB erklärt.

Themen

- Dokumente abfragen
- Plan abfragen
- Erläutern Sie die Ergebnisse
- Abfrageplaner v2
- Abfrageplaner v3
- Abfragen von Geodaten mit Amazon DocumentDB
- Teilweiser Index
- Textsuche mit Amazon DocumentDB durchführen

Dokumente abfragen

Manchmal müssen Sie möglicherweise den Bestand Ihres Online-Shops nachschlagen, damit Kunden das Angebot sehen und kaufen können. Die Abfrage einer Sammlung ist relativ einfach, unabhängig davon, ob Sie alle Dokumente in der Sammlung haben möchten oder nur die Dokumente, die ein bestimmtes Kriterium erfüllen.

Verwenden Sie die Operation `find()`, um Dokumente abzufragen. Der Befehl `find()` hat einen einzigen Dokumentenparameter, der die Kriterien für die Auswahl der zurückzugebenden Dokumente definiert. Die Ausgabe von `find()` ist ein Dokument, das als einzelne Textzeile ohne Zeilenumbrüche formatiert ist. Um das Ausgabedokument für eine bessere Lesbarkeit zu formatieren, verwenden Sie `find().pretty()`. Alle Beispiele in diesem Thema verwenden `.pretty()` zum Formatieren der Ausgabe.

In den folgenden Codebeispielen werden die vier Dokumente verwendet, die Sie in den beiden vorangegangenen Übungen in die `example` Sammlung eingefügt haben — `insertOne()` und `insertMany()` die sich in Dokumente hinzufügen befinden.

Themen

- Alle Dokumente in einer Sammlung werden abgerufen
- Dokumente werden abgerufen, die einem Feldwert entsprechen
- Dokumente werden abgerufen, die einem eingebetteten Dokument entsprechen
- Abrufen von Dokumenten, die einem Feldwert in einem eingebetteten Dokument entsprechen
- Dokumente werden abgerufen, die einem Array entsprechen
- Abrufen von Dokumenten, die einem Wert in einem Array entsprechen
- Dokumente mithilfe von Operatoren abrufen

Alle Dokumente in einer Sammlung werden abgerufen

Um alle Dokumente in Ihrer Sammlung abzurufen, verwenden Sie die Operation `find()` mit einem leeren Abfragedokument.

Die folgende Abfrage gibt alle Dokumente der Sammlung `example` zurück.

```
db.example.find( {} ).pretty()
```

Dokumente werden abgerufen, die einem Feldwert entsprechen

Um alle Dokumente abzurufen, die mit einem Feld und einem Wert übereinstimmen, verwenden Sie die Operation `find()` mit einem Abfragedokument, das die entsprechenden Felder und Werte identifiziert.

Bei Verwendung der vorangegangenen Dokumente gibt diese Abfrage alle Dokumente zurück, bei denen das Feld "Item" (Element) "Pen" (Stift) entspricht.

```
db.example.find( { "Item": "Pen" } ).pretty()
```

Dokumente werden abgerufen, die einem eingebetteten Dokument entsprechen

Um alle Dokumente zu suchen, die mit einem eingebetteten Dokument übereinstimmen, verwenden Sie die Operation `find()` mit einem Abfragedokument, in dem der Name des eingebetteten Dokuments sowie alle Felder und Werte für dieses eingebettete Dokument angegeben werden.

Beim Vergleichen mit einem eingebetteten Dokument muss das eingebettete Dokument denselben Namen haben wie in der Abfrage. Zudem müssen die Felder und Werte im eingebetteten Dokument mit der Abfrage übereinstimmen.

Die folgende Abfrage gibt nur das Dokument "Poster Paint" zurück. Dies liegt daran, dass "Pen" über verschiedene Werte für "`OnHand`" und "`MinOnHand`" verfügt und "Spray Paint" ein weiteres Feld (`OrderQnty`) als das Abfragedokument besitzt.

```
db.example.find({"Inventory": {
    "OnHand": 47,
    "MinOnHand": 50 } } ).pretty()
```

Abrufen von Dokumenten, die einem Feldwert in einem eingebetteten Dokument entsprechen

Um alle Dokumente zu suchen, die mit einem eingebetteten Dokument übereinstimmen, verwenden Sie die Operation `find()` mit einem Abfragedokument, in dem der Name des eingebetteten Dokuments sowie alle Felder und Werte für dieses eingebettete Dokument angegeben werden.

Aufgrund der vorangegangenen Dokumente verwendet die folgende Abfrage "Punktnotation", um das eingebettete Dokument und die Felder von Interesse anzugeben. Jedes Dokument, das damit übereinstimmt, wird zurückgegeben, unabhängig davon, welche anderen Felder im eingebetteten Dokument vorhanden sind. Die Abfrage gibt "Poster Paint" und "Spray Paint" zurück, weil sie beide den angegebenen Feldern und Werten entsprechen.

```
db.example.find({"Inventory.OnHand": 47, "Inventory.MinOnHand": 50 }).pretty()
```

Dokumente werden abgerufen, die einem Array entsprechen

Um alle Dokumente zu finden, die einem Array entsprechen, verwenden Sie die Operation `find()` mit dem Namen des Arrays, an dem Sie interessiert sind, und allen Werten in diesem Array. Die Abfrage gibt alle Dokumente zurück, in denen sich ein Array mit diesem Namen befindet und in denen die Array-Werte identisch sind und die gleiche Reihenfolge wie in der Abfrage aufweisen.

Die folgende Abfrage gibt nur "Pen" zurück, da "Poster Paint" über eine zusätzlichen Farbe (White) verfügt und die Farben in "Spray Paint" in einer anderen Reihenfolge vorliegen.

```
db.example.find( { "Colors": ["Red","Green","Blue","Black"] } ).pretty()
```

Abrufen von Dokumenten, die einem Wert in einem Array entsprechen

Um alle Dokumente mit einem bestimmten Array-Wert zu finden, verwenden Sie die Operation `find()` mit dem Namen und Wert des Arrays, an dem Sie interessiert sind.

```
db.example.find( { "Colors": "Red" } ).pretty()
```

Bei der vorherigen Operation werden alle drei Dokumente zurückgegeben, da jedes davon ein Array mit dem Namen `Colors` und den Wert "`Red`" irgendwo im Array besitzt. Wenn Sie den Wert "`White`" angeben, gibt die Abfrage nur "Poster Paint" zurück.

Dokumente mithilfe von Operatoren abrufen

Die folgende Abfrage gibt alle Dokumente zurück, in denen der Wert "`Inventory.OnHand`" kleiner als 50 ist.

```
db.example.find(
        { "Inventory.OnHand": { $lt: 50 } } )
```

Eine Liste der unterstützten Abfrageoperatoren finden Sie unter Abfrage- und Projektionsoperatoren.

Plan abfragen

Wie kann ich die **`ExecutionStats`** für einen Abfrageplan sehen?

Wenn Sie ermitteln, warum eine Abfrage langsamer als erwartet ausgeführt wird, kann es hilfreich sein zu verstehen, was die `executionStats` für den Abfrageplan sind. Die `executionStats` geben die Anzahl der Dokumente an, die von einer bestimmten Stufe zurückgegeben wurden (`nReturned`), die in jeder Stufe verbrachte Ausführungszeit (`executionTimeMillisEstimate`) und die Zeit, die zum Generieren eines Abfrageplans benötigt wird (`planningTimeMillis`). Sie können die zeitintensivsten Phasen Ihrer Abfrage bestimmen, um Ihre Optimierungsbemühungen auf der Ausgabe von zu konzentrieren`executionStats`, wie in den folgenden Abfragebeispielen gezeigt. Der `executionStats`-Parameter unterstützt derzeit keine `update`- und `delete`-Befehle.

> **Note**
>
> Amazon DocumentDB emuliert die MongoDB 3.6-API auf einer speziell entwickelten Datenbank-Engine, die ein verteiltes, fehlertolerantes, selbstheilendes Speichersystem verwendet. Daher können sich die Abfragepläne und die Ausgabe von zwischen Amazon DocumentDB und MongoDB unterscheiden. `explain()` Kunden, die die Kontrolle über ihren Abfrageplan wünschen, können den `$hint`-Operator verwenden, um die Auswahl eines bevorzugten Indexes zu erzwingen.

Führen Sie die Abfrage, die Sie verbessern möchten, unter dem Befehl `explain()` wie folgt aus.

```
db.runCommand({explain: {query document}}).
explain("executionStats").executionStats;
```

Im Folgenden finden Sie eine Beispieloperation.

```
db.fish.find({}).limit(2).explain("executionStats");
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
    "queryPlanner" : {
        "plannerVersion" : 1,
        "namespace" : "test.fish",
        "winningPlan" : {
            "stage" : "SUBSCAN",
            "inputStage" : {
                "stage" : "LIMIT_SKIP",
                "inputStage" : {
                    "stage" : "COLLSCAN"
                }
            }
        }
    },
    "executionStats" : {
        "executionSuccess" : true,
        "executionTimeMillis" : "0.063",
        "planningTimeMillis" : "0.040",
        "executionStages" : {
            "stage" : "SUBSCAN",
            "nReturned" : "2",
            "executionTimeMillisEstimate" : "0.012",
            "inputStage" : {
                "stage" : "LIMIT_SKIP",
                "nReturned" : "2",
                "executionTimeMillisEstimate" : "0.005",
                "inputStage" : {
                    "stage" : "COLLSCAN",
                    "nReturned" : "2",
                    "executionTimeMillisEstimate" : "0.005"
                }
```

```
            }
        }
    },
    "serverInfo" : {
        "host" : "enginedemo",
        "port" : 27017,
        "version" : "3.6.0"
    },
    "ok" : 1
}
```

Wenn Sie nur die `executionStats` aus der obigen Abfrage sehen möchten, können Sie den folgenden Befehl verwenden. Bei kleinen Sammlungen kann sich der Amazon DocumentDB-Abfrageprozessor dafür entscheiden, keinen Index zu verwenden, wenn die Leistungssteigerungen vernachlässigbar sind.

```
db.fish.find({}).limit(2).explain("executionStats").executionStats;
```

Plan-Cache abfragen

Um die Leistung zu optimieren und die Planungsdauer zu reduzieren, speichert Amazon DocumentDB Abfragepläne intern im Cache. Dadurch können Abfragen mit derselben Form direkt mithilfe eines zwischengespeicherten Plans ausgeführt werden.

Dieses Zwischenspeichern kann jedoch manchmal zu einer zufälligen Verzögerung für dieselbe Abfrage führen. Beispielsweise kann eine Abfrage, deren Ausführung in der Regel eine Sekunde dauert, gelegentlich zehn Sekunden dauern. Das liegt daran, dass die Reader-Instanz im Laufe der Zeit verschiedene Formen der Abfrage zwischengespeichert hat und dadurch Speicherplatz verbraucht hat. Wenn Sie diese zufällige Langsamkeit feststellen, müssen Sie keine Aktion ausführen, um den Speicher freizugeben. Das System verwaltet die Speichernutzung für Sie, und sobald der Speicher einen bestimmten Schwellenwert erreicht, wird er automatisch freigegeben.

Erläutern Sie die Ergebnisse

Wenn Sie Informationen zu Abfrageplänen zurückgeben möchten, unterstützt Amazon DocumentDB den Ausführlichkeitsmodus. `queryPlanner` Die `explain` Ergebnisse geben den vom Optimierer ausgewählten Abfrageplan in einem Format zurück, das dem folgenden ähnelt:

```
{
```

```
   "queryPlanner" : {
      "plannerVersion" : <int>,
      "namespace" : <string>,
      "winningPlan" : {
         "stage" : <STAGE1>,
         ...
         "inputStage" : {
            "stage" : <STAGE2>,
            ...
            "inputStage" : {
               ...
            }
         }
      }
   }
}
```

In den folgenden Abschnitten werden allgemeine `explain` Ergebnisse definiert.

Themen

- Scan- und Filterphase
- Indexüberschneidung
- Vereinigung des Indexes
- Mehrfacher Index intersection/union
- Zusammengesetzter Index
- Phase sortieren
- Gruppenphase

Scan- und Filterphase

Der Optimierer kann einen der folgenden Scans wählen:

COLLSCAN

Diese Phase ist ein sequentieller Sammlungsscan.

```
{
    "stage" : "COLLSCAN"
}
```

IST SCAN

In dieser Phase werden die Indexschlüssel gescannt. Der Optimierer kann das Dokument innerhalb dieser Phase abrufen, was dazu führen kann, dass später ein FETCH-Schritt angehängt wird.

```
db.foo.find({"a": 1})
{
    "stage" : "IXSCAN",
    "direction" : "forward",
    "indexName" : <idx_name>
}
```

FETCH

Wenn der Optimierer Dokumente in einer anderen Phase als IXSCAN abgerufen hat, enthält das Ergebnis einen FETCH-Schritt. Beispielsweise kann die obige IXSCAN-Abfrage zu einer Kombination aus FETCH- und IXSCAN-Schritten führen:

```
db.foo.find({"a": 1})
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXSCAN",
        "indexName" : <idx_name>
    }
}
```

IXONLYSCAN scannt nur den Indexschlüssel. Beim Erstellen zusammengesetzter Indizes wird FETCH nicht vermieden.

Indexüberschneidung

IST UND

Amazon DocumentDB kann einen IXAND-Schritt mit einem InputStages-Array von IXSCAN enthalten, wenn es die Indexüberschneidung nutzen kann. Zum Beispiel könnten wir eine Ausgabe wie die folgende sehen:

```
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXAND",
        "inputStages" : [
            {
                "stage" : "IXSCAN",
                "indexName" : "a_1"
            },
            {
                "stage" : "IXSCAN",
                "indexName" : "b_1"
            }
        ]
    }
}
```

Vereinigung des Indexes

IXOR

Ähnlich wie bei der Indexüberschneidung kann Amazon DocumentDB eine `IXOR` Stufe mit einem `inputStages` Array für den `$or` Operator enthalten.

```
db.foo.find({"$or": [{"a": {"$gt": 2}}, {"b": {"$lt": 2}}]})
```

Für die obige Abfrage könnte die Explain-Ausgabe wie folgt aussehen:

```
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXOR",
        "inputStages" : [
            {
                "stage" : "IXSCAN",
                "indexName" : "a_1"
            },
            {
                "stage" : "IXSCAN",
                "indexName" : "b_1"
```

```
            }
        ]
    }
}
```

Mehrfacher Index intersection/union

Amazon DocumentDB kann mehrere Indexüberschneidungs- oder Union-Stufen miteinander kombinieren und dann das Ergebnis abrufen. Beispiel:

```
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXOR",
        "inputStages" : [
            {
                "stage" : "IXSCAN",
                ...
            },
            {
                "stage" : "IXAND",
                "inputStages" : [
                    {
                        "stage" : "IXSCAN",
                        ...
                    },
                    {
                        "stage" : "IXSCAN",
                        ...
                    }
                ]
            }
        ]
    }
}
```

Die Verwendung von Indexschnittstellen oder Union-Stufen wird durch den Indextyp (spärlich, zusammengesetzt usw.) nicht beeinflusst.

Zusammengesetzter Index

Die Verwendung des zusammengesetzten Amazon DocumentDB-Indexes ist nicht auf die ersten Teilmengen indizierter Felder beschränkt. Es kann den Index mit dem Suffixteil verwenden, dies ist jedoch möglicherweise nicht sehr effizient.

Der zusammengesetzte Index von `{ a: 1, b: -1 }` kann beispielsweise alle drei unten aufgeführten Abfragen unterstützen:

```
db.orders.find( { a: 1 } )

db.orders.find( { b: 1 } )

db.orders.find( { a: 1, b: 1 } )
```

Phase sortieren

Wenn es einen Index für die angeforderten Sortierschlüssel gibt, kann Amazon DocumentDB den Index verwenden, um die Reihenfolge abzurufen. In diesem Fall enthält das Ergebnis keine `SORT` Phase, sondern eine `IXSCAN` Phase. Wenn der Optimierer eine einfache Sortierung bevorzugt, wird er eine Stufe wie diese enthalten:

```
{
    "stage" : "SORT",
    "sortPattern" : {
        "a" : 1,
        "b" : -1
    }
}
```

Gruppenphase

Amazon DocumentDB unterstützt zwei verschiedene Gruppenstrategien:

- `SORT_AGGREGATE`: Aggregat auf der Festplatte sortieren.
- `HASH_AGGREGATE`: Hash-Aggregat im Speicher.

Abfrageplaner v2

Der neue Abfrageplaner für Amazon DocumentDB (Planer-Version 2.0) bietet erweiterte Funktionen zur Abfrageoptimierung und eine verbesserte Leistung. Die Planner-Version 2.0 für Amazon DocumentDB 5.0 bietet eine bis zu 10-fache Leistungsverbesserung gegenüber der Vorgängerversion, wenn Sie `update` AND-Operatoren mit `find` Indizes verwenden. Leistungsverbesserungen sind in erster Linie auf die Verwendung optimalerer Indexpläne und die Aktivierung der Unterstützung von Indexscans für Operatoren wie Negationsoperatoren (`$neq`,`$nin`) und Nested zurückzuführen. `$elementMatch` Abfragen von Planner Version 2.0 werden dank besserer Kostenschätzungstechniken, optimierter Algorithmen und verbesserter Stabilität schneller ausgeführt. Planner Version 2.0 unterstützt auch Plan-Cache-Filter-APIs, wodurch die Stabilität des Planers verbessert wird. Mit dieser Funktion bietet Amazon DocumentDB 5.0 jetzt die Möglichkeit, aus den verschiedenen Versionen des Abfrageplaners auszuwählen.

Topics

- Voraussetzungen
- Auswahl der Planner-Version 2.0 als Standard-Abfrageplaner
- Bewährte Methoden
- Einschränkungen
- Verbesserungen beim Suchen und Aktualisieren von Operatoren
- Planen Sie Cache-Filter
- Mögliche Verhaltensunterschiede zwischen Planner-Version 1.0, 2.0 und MongoDB
- Planner Version 2.0 überbrückt Verhaltenslücken mit MongoDB

Voraussetzungen

Die folgenden Voraussetzungen gelten für die Planner-Version 2.0:

- Planner Version 2.0 ist überall dort verfügbar AWS-Regionen , wo Engine-Version 5.0 verfügbar ist.
- Um sich für die Verwendung von Version 2.0 als Standard-Abfrageplaner zu entscheiden, muss Ihr Cluster auf dem Engine-Patch Version 3.0.15902 oder höher der Amazon DocumentDB-Version 5.0 installiert sein. Schritte zur Aktualisierung auf den neuesten Patch für die Engine-Version finden Sie unter. Durchführung eines Patch-Updates für die Engine-Version eines Clusters
- Um Planner Version 2.0 als Standard-Abfrageplaner festzulegen, benötigen Sie IAM-Berechtigungen, um Cluster-Parametergruppen zu aktualisieren.

- Für Amazon DocumentDB 8.0 ist Planner Version 3.0 der Standard-Abfrageplaner. Details dazu finden Sie unter Abfrageplaner v3.

Auswahl der Planner-Version 2.0 als Standard-Abfrageplaner

Gehen Sie wie folgt vor, um 2.0 in der Konsole oder CLI als Standard-Abfrageplaner auszuwählen:

- Folgen Sie den Schritten unterÄnderung der Amazon DocumentDB-Cluster-Parameter, um die Parametergruppe Ihres Clusters zu ändern.
- Ändern Sie den Wert für den Parameter mit dem Titel „PlannerVersion“ auf 2.0, was auf die Planerversion 2.0 hinweist.
- Wählen Sie Sofort anwenden aus (wenn Sie Beim Neustart anwenden auswählen, wird die Auswahl bis zum nächsten Neustart des Clusters unwirksam).

Bewährte Methoden

Verwenden Sie bei der Anwendung von Planner Version 2.0 die folgenden bewährten Methoden, um die erwarteten Ergebnisse zu erzielen:

- Wählen Sie in einem globalen Cluster den gleichen `plannerVersion` Wert (1,0 oder 2,0) in den Cluster-Parametergruppen für beide Regionen aus. Beachten Sie, dass die Auswahl unterschiedlicher Planerversionen in primären und sekundären Regionen zu einem inkonsistenten Abfrageverhalten und einer inkonsistenten Leistung führen kann.
- Die Aktualisierung auf Planner-Version 2.0 während eines geplanten Wartungsfensters oder in Zeiten mit reduziertem Verkehrsaufkommen ist am wenigsten störend, da es zu erhöhten Fehlerraten kommen kann, wenn die Planner-Version geändert wird, während Workloads aktiv ausgeführt werden.
- Planner Version 2.0 funktioniert am optimalsten mit der MongoDB-Shell-Version 5.0.

Einschränkungen

Die folgenden Einschränkungen gelten für die Planner-Version 2.0:

- Planner-Version 2.0 wird in elastischen Clustern nicht unterstützt. Diese werden auf Planner-Version 1.0 zurückgreifen.

- Planner-Version 2.0 wird für Aggregation und unterschiedliche Befehle nicht unterstützt. Diese werden auf Planner-Version 1.0 zurückgegriffen.
- Für einige Abfrage-Shapes kann kein Plan-Cache-Filter festgelegt werden. Die aktuelle Liste finden Sie unterUnterstützte Abfrage-Shapes.

Verbesserungen beim **Suchen** und **Aktualisieren von** Operatoren

Planner Version 2.0 optimiert grundlegende Operationen wie`find`, `update`, `delete`, und `find-and-modify` Befehle. In den folgenden Abschnitten mit Registerkarten werden die erweiterten Funktionen für Indizes sowie die Verbesserungen der Abfrageleistung mit Planner Version 2.0 beschrieben:

Enhanced index support

- Planner Version 2.0 bietet Indexunterstützung für Negationsoperatoren wie `$nin` `$ne`, `$not {eq}` `$not {in}`, und sowie und und. `$type` `$elemMatch`

```
Sample Document: { "x": 10, "y": [1, 2, 3] }

db.foo.createIndex({ "x": 1, "y": 1 })
db.foo.find({ "x": {$nin: [20, 30] }})
db.foo.find({"x":{ $type: "string" }})

db.foo.createIndex({"x.y": 1})
db.foo.find({"x":{$elemMatch:{"y":{$elemMatch:{"$gt": 3 }}}}})
```

- Planner Version 2.0 verwendet Indizes mit geringer Dichte oder nur teilweise, auch wenn sie im `$exists` Abfrageausdruck nicht vorhanden sind.

```
Sample Document: {"name": "Bob", "email": "example@fake.com" }

Using Planner Version 1.0, you can specify the command as shown below:
db.foo.find({email: "example@fake.com", email: {$exists: true}})

Using Planner Version 2.0, you can specify command without $exists:
db.foo.find({ email: "example@fake.com" })
```

- Planner Version 2.0 verwendet Teilindizes auch dann, wenn die Abfragebedingung nicht genau mit dem Filterausdruck für den Teilindex übereinstimmt.

```
Sample Document: {"name": "Bob", "age": 34}
```

```
db.foo.createIndex({"age":1},{partialFilterExpression:{"age":{$lt:50}}})

With Planner Version 1.0, index is used only when the query condition meets the
 partial
 index filter criterion:
     db.foo.find({"age":{$lt:50}})

With Planner Version 2.0, index is used even when the query condition doesn't meet
 the index
criterion:
db.foo.find({"age":{$lt:30}})
```

- Planner Version 2.0 verwendet einen teilweisen Indexscan mit $elemMatch-Abfragen.

```
Sample Document: {"name": "Bob", "age": [34,35,36]}
db.foo.createIndex({"age":1},{partialFilterExpression:{"age":{$lt:50,$gt:20}}})
db.foo.find({age:{$elemMatch:{$lt:50,$gt:20}}})
```

- Planner Version 2.0 unterstützt Index-Scans für`$regex`, ohne dass Sie den Code `$hint` in Ihrer Anwendung angeben müssen. `$regex`unterstützt den Index nur bei Präfixsuchen.

```
Sample Document: { "x": [1, 2, 3], "y": "apple" }
db.foo.createIndex({ "x": 1, "y": 1 })
db.foo.find({"y":{ $regex: "^a" }})
```

- Planner Version 2.0 verbessert die Leistung von Abfragen mit Indizes mit mehreren Schlüsseln, wobei Gleichheitsbedingungen für Felder mit mehreren Schlüsseln gelten.

```
Sample Document: {"x": [1, 2, 3],
"y": 5}
db.foo.createIndex({"x": 1, "y":1})
db.foo.find({"x": 2,
"y": {$gt: 1}}).limit(1)
```

- Planner Version 2.0 verbessert die Leistung von Abfragen, die mehrere Filter beinhalten, insbesondere bei Sammlungen mit Dokumenten, die größer als 8 KB sind.

```
Sample Document: {"x": 2,
"y": 4,
"z": 9,
"t": 99}
db.foo.find({$and: [{"x": {$gt : 1}, "y": {$gt : 3}, "z": {$lt : 10},
```

```
"t":{$lt : 100}}]})
```

- Planner Version 2.0 verbessert die Leistung von Abfragen, wenn der `$in` Operator mit einem zusammengesetzten Index verwendet wird, indem die Sortierphase entfällt.

```
Sample Document: {"x": 2,
"y": 4,
"z": 9,
"t": 99}
db.foo.createIndex({"x":1, "y":1})
db.foo.find({"x":2,
"y":$in:[1,2,3,4]}).sort({x:1,y:1})
```

Es verbessert auch die Leistung von Abfragen, die Mehrschlüsselindizes mit `$in` Elementen verwenden.

```
Sample Document: {"x": [1, 2, 3]}
db.foo.createIndex({"x": 1})
db.foo.find("x":{$in:[>100 elements]})
```

Query performance improvements

- Planner Version 2.0 verbessert die Leistung von Abfragen, die mehrere Filter beinhalten, insbesondere bei Sammlungen mit Dokumenten, die größer als 8 KB sind.

```
Sample Document: {"x": 2,
"y": 4,
"z": 9,
"t": 99}
db.foo.find({$and: [{"x": {$gt : 1}, "y": {$gt : 3}, "z": {$lt : 10},
"t":{$lt : 100}}]})
```

- Planner Version 2.0 verbessert die Leistung von Abfragen, wenn der `$in` Operator mit einem zusammengesetzten Index verwendet wird, indem die Sortierphase entfällt.

```
Sample Document: {"x": 2,
"y": 4,
"z": 9,
"t": 99}
db.foo.createIndex({"x":1, "y":1})
```

```
db.foo.find({"x":2,
"y":$in:[1,2,3,4]}).sort({x:1,y:1})
```

Es verbessert auch die Leistung von Abfragen, die Multikey-Indizes mit `$in` Elementen verwenden.

```
Sample Document: {"x": [1, 2, 3]}
db.foo.createIndex({"x": 1})
db.foo.find("x":{$in:[>100 elements]})
```

Planen Sie Cache-Filter

Planner Version 2.0 bietet Unterstützung für Plan-Cache-Filter, auch Indexfilter genannt, mit denen Sie die Gruppe von Indizes einschränken können, die der Planer für eine bestimmte Abfrageform berücksichtigt. Filter werden mit einem Datenbankbefehl festgelegt und auf den Server angewendet. Wenn Sie also eine Abfrageregression feststellen, können Sie diese abmildern, ohne Ihren Anwendungscode zu ändern.

In der Planner-Version 2.0 gelten Plan-Cache-Filter für die `count` Befehle `find` und ab Amazon DocumentDB 5.0.2 und 8.0.2 auch für die `update` Befehle, und. `delete` `findAndModify` Für die `aggregate` Befehle `distinct` und ist die Planner-Version 3.0 erforderlich.

Weitere Informationen zur Befehlsreferenz, zu unterstützten Abfrageformen, funktionierenden Beispielen und zur Interaktion von Filtern mit Hinweisen finden Sie unterCache-Filter planen.

Mögliche Verhaltensunterschiede zwischen Planner-Version 1.0, 2.0 und MongoDB

In einigen Randfällen ist es möglich, dass Planner-Version 2.0 Ergebnisse liefert, die leicht von den Ergebnissen in MongoDB abweichen. In diesem Abschnitt werden einige Beispiele für diese Möglichkeiten vorgestellt.

$(update) and $(projection)

- In einigen Fällen verhalten sich `$(update)` `$(projection)` Operatoren in MongoDB möglicherweise anders als in der Planer-Version 1.0 von Amazon DocumentDB. Im Folgenden finden Sie einige Beispiele:

```
db.students_list.insertMany( [ { _id: 5, student_ids: [ 100, 200 ], grades: [ 95,
 100 ], grad_year: [ 2024, 2023 ] } ] )
```

```
db.students_list.updateOne({ student_ids: 100, grades: 100, grad_year: 2024 },
{ $set: { “grad_year.$”: 2025 } }
```

- Planner Version 1.0 — Aktualisiert das Feld 2022
- MongoDB — Aktualisiert das Feld 2022
- Planner Version 2.0 — Aktualisiert das Feld 2021

- ```
  db.col.insert({x:[1,2,3]})
  db.col.update({$and:[{x:1},{x:3}]},{$set:{"x.$":500}})
  ```
  - Planner Version 1.0 — Aktualisiert nach dem Zufallsprinzip das erste übereinstimmende Element
  - MongoDB — Aktualisiert nach dem Zufallsprinzip das erste übereinstimmende Element
  - Planner Version 2.0 — Führt keine Aktualisierungen durch
- ```
  db.col.insert({x:[1,2,3]})
  db.col.find()
  ```
  - Planner Version 1.0 — Wählt das übereinstimmende Element nach dem Zufallsprinzip aus
  - MongoDB — Wählt das übereinstimmende Element nach dem Zufallsprinzip aus
  - Planner Version 2.0 — Trifft keine Auswahl
- ```
  db.col.insert({x:100})
  db.col.update({x:100},{x:100})
  ```
  - Planner-Version 1.0 — Änderung der Anzahl der Änderungen
  - MongoDB — Änderungen an der Anzahl
  - Planner Version 2.0 — Die Anzahl von nModified ändert sich nicht, wenn sie mit demselben Wert aktualisiert wird.
- Wenn der `$(update)` Operator mit verwendet wird`$setOnInsert`, geben Planner Version 1.0 und MongoDB einen Fehler aus, Planner Version 2.0 jedoch nicht.
- Das Umbenennen eines nicht existierenden Felds in Planner Version 2.0 `$field` führt zu einem Fehler, während es in Planner Version 1.0 und MongoDB nicht zu Aktualisierungen kommt.
  ```

Index behavior

- Planner-Version 2.0 gibt einen Fehler aus, wenn sie mit einem ungeeigneten Index angewendet `$hint` wird, wohingegen Planner-Version 1.0 und MongoDB dies nicht tun.

```
// Insert
db.col.insert({x:1})
db.col.insert({x:2})
db.col.insert({x:3})

// Create index on x with partialFilter Expression {x:{$gt:2}}
db.col.createIndex({x:1},{partialFilterExpression:{x:{$gt:2}}})

// Mongodb allows hint on the following queries
db.col.find({x:1}).hint("x_1")
// result is no documents returned because {x:1} is not indexed by the partial
 index
// Without $hint mongo should return {x:1}, thus the difference in result between
 COLSCAN and IXSCAN

DocumentDB will error out when $hint is applied on such cases.
db.col.find({x:1}).hint("x_1")
Error: error: {
    "ok" : 0,
    "operationTime" : Timestamp(1746473021, 1),
    "code" : 2,
    "errmsg" : "Cannot use Hint for this Query. Index is multi key index , partial
 index or sparse index and query is not optimized to use this index."
}

rs0:PRIMARY> db.runCommand({"planCacheSetFilter": "col", "query": { location:
 {$nearSphere: {$geometry: {type: "Point", coordinates: [1, 1]}}}}, "indexes":
 ["name_1"]})
{
    "ok" : 0,
    "operationTime" : Timestamp(1750815778, 1),
    "code" : 303,
    "errmsg" : "Unsupported query shape for index filter $nearSphere"
}
```

- `$near`kann `$hint({“$natural”:1})` in Planner-Version 2.0 nicht verwendet werden.

```
// indexes present are index on x and geo index
```

```
rs0:PRIMARY> db.usarestaurants.getIndexes()
[
    {
        "v" : 4,
        "key" : {
            "_id" : 1
        },
        "name" : "_id_",
        "ns" : "test.usarestaurants"
    },
    {
        "v" : 4,
        "key" : {
            "location" : "2dsphere"
        },
        "name" : "location_2dsphere",
        "ns" : "test.usarestaurants",
        "2dsphereIndexVersion" : 1
    }
]

// Planner Version 2.0 will throw an error when $hint is applied with index "x_1"
rs0:PRIMARY> db.usarestaurants.find({    "location":{       "$nearSphere":{
         "$geometry":{             "type":"Point",             "coordinates":
[                -122.3516,                47.6156             ]          },
          "$minDistance":1,          "$maxDistance":2000       }    } }, {
  "name":1 }).hint({"$natural": 1})
Error: error: {
    "ok" : 0,
    "operationTime" : Timestamp(1746475524, 1),
    "code" : 291,
    "errmsg" : "unable to find index for $geoNear query"
}

// Planner Version 1.0 and MongoDB will not throw an error
 db.usarestaurants.find({    "location":{       "$nearSphere":{
  "$geometry":{             "type":"Point",             "coordinates":[
             -122.3516,                47.6156             ]          },
        "$minDistance":1,          "$maxDistance":2000       }    } }, {
  "name":1 }).hint({"$natural": 1})
{ "_id" : ObjectId("681918e087dadfd99b7f0172"), "name" : "Noodle House" }
```

- Während MongoDB vollständige Regex-Index-Scans unterstützt, unterstützt Planner Version 2.0 den Regex-Index-Scan nur für Präfixfelder.

```
// index on x
db.col.createIndex({x:1})

// index scan is used only for prefix regexes
rs0:PRIMARY> db.col.find({x: /^x/}).explain()
{
    "queryPlanner" : {
        "plannerVersion" : 2,
        "namespace" : "test.col",
        "winningPlan" : {
            "stage" : "IXSCAN",
            "indexName" : "x_1",
            "direction" : "forward",
            "indexCond" : {
                "$and" : [
                    {
                        "x" : {
                            "$regex" : /^x/
                        }
                    }
                ]
            },
            "filter" : {
                "x" : {
                    "$regex" : /^x/
                }
            }
        }
    },
    "indexFilterSet" : false,
    "indexFilterApplied" : false,
    "ok" : 1,
    "operationTime" : Timestamp(1746474527, 1)
}

// COLSCAN is used for non-prefix regexes
rs0:PRIMARY> db.col.find({x: /x$/}).explain()
{
    "queryPlanner" : {
        "plannerVersion" : 2,
```

```
        "namespace" : "test.col",
        "winningPlan" : {
            "stage" : "COLLSCAN",
            "filter" : {
                "x" : {
                    "$regex" : /x$/
                }
            }
        }
    },
    "indexFilterSet" : false,
    "indexFilterApplied" : false,
    "ok" : 1,
    "operationTime" : Timestamp(1746474575, 1)
```

- Es gibt einige inhärente Unterschiede bei der Verwendung von Plan-Cache-Filtern mit Planner-Version 2.0 im Vergleich zu MongoDB. Während die Planer-Version 2.0 die Angabe von „Projektion“ und „Sortierung“ mit Plan-Cache-Filtern nicht unterstützt, tut MongoDB dies. Der MongoDB-Indexfilter befindet sich jedoch nur im Arbeitsspeicher und geht nach dem Neustart verloren. In Planner Version 2.0 werden Indexfilter auch bei Neustarts und Patches beibehalten.

Others

- Das Format der DML-Auditprotokolle bei Verwendung der Planner-Version 2.0 unterscheidet sich geringfügig von dem der Planner-Version 1.0.

```
command -  db.col.find({x:1})

************** Audit logs generated ******************

// v1 format for dml audit logs
{"atype":"authCheck","ts":1746473479983,"timestamp_utc":"2025-05-05
 19:31:19.983","remote_ip":"127.0.0.1:47022","users":
[{"user":"serviceadmin","db":"test"}],"param":
{"command":"find","ns":"test.col","args":{"batchSize":101,"filter":
{"x":1},"find":"col","limit":18446744073709551615,"lsid":
{"id":{"$binary":"P6RCGz9ZS4iWBSSHWXW15A==","$type":"4"},"uid":
{"$binary":"6Jo8PisnEi3dte03+pJFjdCyn/5cGQL8V2KqaoWsnk8=","$type":"0"}},"maxScan":18446744

// v2 formal for dml audit logs
{"atype":"authCheck","ts":1746473583711,"timestamp_utc":"2025-05-05
 19:33:03.711","remote_ip":"127.0.0.1:37754","users":
```

```
[{"user":"serviceadmin","db":"test"}],"param":
{"command":"find","ns":"test.col","args":{"find":"col","filter":{"x":1},"lsid":
{"id":{"$binary":"nJ88TGCSSd
+BeD2+ZtrhQg==","$type":"4"}},"$db":"test"},"result":0}}
```

- Die Indexbedingung als Teil des Explain-Plans:

```
rs0:PRIMARY> db.col.createIndex({index1:1})
{
 "createdCollectionAutomatically" : false,
 "numIndexesBefore" : 1,
 "numIndexesAfter" : 2,
 "ok" : 1,
 "operationTime" : Timestamp(1761149251, 1)
}
```

Planner Version 2.0 erklärt die Planausgabe mit Anzeige der Indexbedingung und des Filters:

```
rs0:PRIMARY> db.col.find({$and:[{price:{$eq:300}},{item:
{$eq:"apples"}}]}).explain()
{
 "queryPlanner" : {
  "plannerVersion" : 2,
  "namespace" : "test.col",
  "winningPlan" : {
   "stage" : "IXSCAN",
   "indexName" : "price_1",
   "direction" : "forward",
   "indexCond" : {
    "$and" : [
     {
      "price" : {
       "$eq" : 300
      }
     }
    ]
   },
   "filter" : {
    "$and" : [
     {
      "item" : {
       "$eq" : "apples"
      }
```

```
     }
    ]
   }
  }
 },
 "indexFilterSet" : false,
 "indexFilterApplied" : false,
 "ok" : 1,
 "operationTime" : Timestamp(1761149497, 1)
}
```

Planner Version 1.0 erklärt die Planausgabe:

```
rs0:PRIMARY> db.col.find({$and:[{price:{$eq:300}},{item:
{$eq:"apples"}}]}).explain()
{
 "queryPlanner" : {
  "plannerVersion" : 1,
  "namespace" : "test.col",
  "winningPlan" : {
   "stage" : "IXSCAN",
   "indexName" : "price_1",
   "direction" : "forward"
  }
 },
 "ok" : 1,
 "operationTime" : Timestamp(1761149533, 1)
}
```

Planner Version 2.0 überbrückt Verhaltenslücken mit MongoDB

Es gibt einige Bereiche, in denen Planner Version 2.0 die Verhaltenslücken von MongoDB schließt:

- Die Planner-Version 2.0 ermöglicht die numerische Indexsuche in abgeflachten Arrays für: `$elemMatch`

```
doc: {"x" : [ [ { "y" : 1 } ] ] }

// Planner Version 2 and mongo
> db.bar.find({"x.0": {$elemMatch: {y: 1}}})
{ "_id" : ObjectId("68192947945e5846634c455a"), "x" : [ [ { "y" : 1 } ] ] }
```

```
> db.bar.find({"x": {$elemMatch: {"0.y": 1}}})
{ "_id" : ObjectId("68192947945e5846634c455a"), "x" : [ [ { "y" : 1 } ] ] }

//Whereas Planner Version 1 wouldn't return any results.
> db.bar.find({"x.0": {$elemMatch: {y: 1}}})
> db.bar.find({"x": {$elemMatch: {"0.y": 1}}})
```

- Während Planner Version 1.0 Zeichenketten in der Projektion ausschloss, entspricht das Verhalten von Planner Version 2.0 dem von MongoDB und behandelt sie als Literalwerte“

```
// Planner V2/ MongoDB
> db.col.find()
{ "_id" : ObjectId("681537738aa101903ed2fe05"), "x" : 1, "y" : 1 }
> db.col.find({},{x:"string"})
{ "_id" : ObjectId("681537738aa101903ed2fe05"), "x" : "string" }

// Planner V1 treats strings as exclude in projection
rs0:PRIMARY> db.col.find()
{ "_id" : ObjectId("68153744d42969f11d5cca72"), "x" : 1, "y" : 1 }
rs0:PRIMARY> db.col.find({},{x:"string"})
{ "_id" : ObjectId("68153744d42969f11d5cca72"), "y" : 1 }
```

- Planner-Version 2.0 erlaubt wie MongoDB keine Projektion auf dieselben Felder „x“ und „x.a“:

```
// Planner version 2/MongoDB will error out
> db.col.find()
{ "_id" : ObjectId("68153da2012265816bc9ba23"), "x" : [ { "a" : 1 }, 3 ] }
 db.col.find({},{"x.a":1,"x":1}) // error

// Planner Version 1 does not error out
db.col.find()
{ "_id" : ObjectId("68153da2012265816bc9ba23"), "x" : [ { "a" : 1 }, 3 ] }

db.col.find({},{"x.a":1,"x":1})
{ "_id" : ObjectId("68153d60143af947c720d099"), "x" : [ { "a" : 1 }, 3 ] }
```

- Planner-Version 2.0 ermöglicht wie MongoDB die Projektion auf Unterdokumente:

```
// Planner Version2/MongoDB supports projections on subdocuments
 db.col.find()
{ "_id" : ObjectId("681542d8f35ace71f0a50004"), "x" : [ { "y" : 100 } ] }
> db.col.find({},{"x":{"y":1}})
{ "_id" : ObjectId("681542b7a22d548e4ac9ddea"), "x" : [ { "y" : 100 } ] }
```

```
// Planner V1 throws error if projection is subdocument
 db.col.find()
{ "_id" : ObjectId("681542d8f35ace71f0a50004"), "x" : [ { "y" : 100 } ] }
rs0:PRIMARY> db.col.find({},{"x":{"y":1}})
Error: error: {
    "ok" : 0,
    "operationTime" : Timestamp(1746223914, 1),
    "code" : 2,
    "errmsg" : "Unknown projection operator y"
}
```

- In der Planner-Version 2.0 unterstützt die Projektion wie bei MongoDB keine Felder nach dem Operator: $

```
// Mongo and Planner Version 2 will error out
  db.col.find()
{ "_id" : ObjectId("68155fa812f843439b593f3f"), "x" : [ { "a" : 100 } ] }
 db.col.find({"x.a":100},{"x.$.a":1}) - // error

// v1 will not error out
 db.col.find()
{ "_id" : ObjectId("68155fa812f843439b593f3f"), "x" : [ { "a" : 100 } ] }
db.col.find({"x.a":100},{"x.$.a":1})
{ "_id" : ObjectId("68155dee13b051d58239cd0a"), "x" : [ { "a" : 100 } ] }
```

- Planner-Version 2.0 ermöglicht `$hint` wie MongoDB die Verwendung von:

```
// v1 will error out on $hint if there are no filters
db.col.find({}).hint("x_1")
Error: error: {
    "ok" : 0,
    "operationTime" : Timestamp(1746466616, 1),
    "code" : 2,
    "errmsg" : "Cannot use Hint for this Query. Index is multi key index , partial
 index or sparse index and query is not optimized to use this index."
}


// Mongo and Planner Version 2 will allow $hint usage
 db.col.find({}).hint("x_1")
{ "_id" : ObjectId("6818f790d5ba9359d68169cf"), "x" : 1 }
```

Abfrageplaner v3

Planner Version 3 in Amazon DocumentDB 8.0 unterstützt 21 Aggregationsstufen, darunter 6 neue Stufen. Planner V3 bietet integrierte Unterstützung für verschiedene Befehle. Es bietet eine bis zu 2-fache Gesamtleistungsverbesserung gegenüber Planner v2 in Amazon DocumentDB 5.0. Alle neuen Funktionen und Operatoren in Amazon DocumentDB 8.0 sind mit Planner v3 kompatibel. Zu den neuen Aggregationsstufen in Amazon DocumentDB 8.0, die von Planner v3 unterstützt werden, gehören $replaceWith, $vectorSearch, $merge, $set, $unset, $bucket. Planner v3 unterstützt auch neue Funktionen und Operatoren in Amazon DocumentDB 8.0, darunter Collation, Views, $merge, $pow, $rand, $dateTrunc, $date, $date. ToParts FromParts

Topics

- Voraussetzungen
- Auswahl der Planner-Version 3.0 als Standard-Abfrageplaner
- Bewährte Methoden
- Einschränkungen
- Verbesserungen an aggregierten und unterschiedlichen Operatoren
- Cache-Filter planen
- Mögliche Verhaltensunterschiede zwischen Planner-Version 1.0, 3.0 und MongoDB

Voraussetzungen

Die folgenden Voraussetzungen gelten für die Planner-Version 3.0:

- Planner Version 3.0 ist überall dort verfügbar AWS-Regionen , wo Engine-Version 8.0 verfügbar ist.
- Planner Version 3.0 ist der Standard-Abfrageplaner, wenn Engine-Version 8.0 ausgewählt ist.

Auswahl der Planner-Version 3.0 als Standard-Abfrageplaner

Wenn Sie Ihren Standard-Abfrageplaner in Amazon DocumentDB 8.0 geändert haben und zu Planner v3 zurückkehren müssen, können Sie dies über die Konsole oder CLI tun:

- Folgen Sie den Schritten unterÄnderung der Amazon DocumentDB-Cluster-Parameter, um die Parametergruppe Ihres Clusters zu ändern.
- Ändern Sie den Wert für den Parameter mit dem Titel „PlannerVersion“ auf 3.0, was auf die Planerversion 3.0 hinweist.

- Wählen Sie Sofort anwenden (wenn Sie Beim Neustart anwenden auswählen, wird die Auswahl bis zum nächsten Neustart des Clusters unwirksam).

Bewährte Methoden

Verwenden Sie bei der Anwendung von Planner Version 3.0 die folgenden bewährten Methoden, um die erwarteten Ergebnisse zu erzielen:

- Wählen Sie in einem globalen Cluster den gleichen `plannerVersion` Wert (1,0 oder 2,0 oder 3,0) in den Cluster-Parametergruppen für beide Regionen aus. Beachten Sie, dass die Auswahl unterschiedlicher Planerversionen in primären und sekundären Regionen zu einem inkonsistenten Abfrageverhalten und einer inkonsistenten Leistung führen kann.
- Die Aktualisierung auf Planner-Version 3.0 während eines geplanten Wartungsfensters oder in Zeiten mit reduziertem Verkehrsaufkommen ist am wenigsten störend, da es zu erhöhten Fehlerraten kommen kann, wenn die Planner-Version geändert wird, während Workloads aktiv ausgeführt werden.

Einschränkungen

Die folgenden Einschränkungen gelten für Planner Version 3.0:

- Planner-Version 3.0 wird in elastischen Clustern nicht unterstützt. Diese werden auf Planner-Version 1.0 zurückgreifen.

- Während Planner v1 die Verwendung von 'PlanHint' ermöglicht, um sicherzustellen, dass ein bestimmter Abfrageplan vom Abfrageoptimierer ausgewählt wird, erlaubt Planner v3 die Verwendung von 'PlanHint' nicht und stützt sich auf interne Optimierungen, um den besten Plan für die angegebene Abfrage auszuwählen.

Verbesserungen an aggregierten und unterschiedlichen Operatoren

Planner Version 3.0 führt Verbesserungen in allen $aggregate-Stufen und beim $distinct-Befehl ein. Im Folgenden sind einige der bemerkenswertesten Verbesserungen aufgeführt..

- Der Planer verschiebt $match-Phasen, wenn möglich, früher in der Pipeline, wodurch die Anzahl der Dokumente reduziert wird, die in den nachfolgenden Phasen verarbeitet werden.

```
//Planner v1
db.orders.aggregate([
  { $project: { customerId: 1, orderDate: 1, totalAmount: 1 } },
  { $match: { customerId: { $gt: 1000 } } }
])

// Planner v3 pulls up the match since customerId exists in original documents
// Optimized internally as:
db.orders.aggregate([
  { $match: { customerId: { $gt: 1000 } } },  // Pulled up before project
  { $project: { customerId: 1, orderDate: 1, totalAmount: 1 } }
])
```

- Der Planer kombiniert automatisch die Schritte $lookup und $unwind, wenn sie dasselbe Feld bearbeiten. Dadurch werden zwischengeschaltete Datenverarbeitungsvorgänge reduziert und die Leistung verbessert.

```
                Example query:
//Planner v1
db.orders.aggregate([
  { $lookup: { from: "products", localField: "productId", foreignField: "_id", as:
 "productInfo" } },
  { $unwind: "$productInfo" },
  { $project: { orderDate: 1, "productInfo.name": 1, "productInfo.price": 1 } }
])

// Planner version 3.0 optimizes this internally by coalescing the $lookup and
 $unwind stages
```

- In Planner Version 3.0 wird eine neue Strategie zur Ausführung von Distinct Scans eingeführt, die die Leistung bei unterschiedlichen Operationen mit Indizes mit niedriger Kardinalität erheblich verbessert.

```
                Example query:
//// If there is a low cardinality index on "category", you may see a query plan like
 below
db.explain().products.distinct("category")
```

```
"queryPlanner" : {
        "plannerVersion" : 3,
        "namespace" : "db.products",
        "winningPlan" : {
                "stage" : "AGGREGATE",
                "inputStage" : {
                        "stage" : "DISTINCT_SCAN",
                        "inputStage" : {
                                "stage" : "IXONLYSCAN",
                                "indexName" : "category_1",
                                "direction" : "forward"
                        }
                }
        }
}
```

Cache-Filter planen

Planner Version 3.0 unterstützt Plan-Cache-Filter, auch Indexfilter genannt, mit denen Sie die Gruppe von Indizes einschränken können, die der Planer für eine bestimmte Abfrageform berücksichtigt. Filter werden mit einem Datenbankbefehl festgelegt und auf den Server angewendet. Wenn Sie also eine Abfrageregression feststellen, können Sie diese abmildern, ohne Ihren Anwendungscode zu ändern.

Planner Version 3.0 wendet Plan-Cache-Filter auf mehr Befehle an als Planner Version 2.0:

- Für den `distinct` Befehl ist Planner Version 3.0 erforderlich.
- Für den `aggregate` Befehl sind Planner Version 3.0 und Amazon DocumentDB 8.0.2 erforderlich. Ein Filter wird angewendet, wenn die Pipeline die Sammlung durch eine führende `$match` Phase liest.
- Ein Filter, der für die ausländische Sammlung einer `$lookup` `$graphLookup` OR-Stufe gesetzt wird, steuert den Index, der zum Scannen dieser fremden Sammlung verwendet wird. Dazu sind auch Planner Version 3.0 und Amazon DocumentDB 8.0.2 erforderlich.

Weitere Informationen zur Befehlsreferenz, zu unterstützten Abfrageformen, funktionierenden Beispielen und zur Interaktion von Filtern mit Hinweisen finden Sie unter. Cache-Filter planen

Mögliche Verhaltensunterschiede zwischen Planner-Version 1.0, 3.0 und MongoDB

In einigen Randfällen ist es möglich, dass Planner-Version 3.0 Ergebnisse liefert, die leicht von der Planer-Version 1.0 abweichen. In diesem Abschnitt werden einige Beispiele für diese Möglichkeiten vorgestellt.

Feature Differences

- Bei einer leeren Sammlung oder wenn frühere Stufen wie Match alle Dokumente filtern, gibt Planner v1 die Ausgabe „field“ :0 aus. Planner v3 und MongoDB geben keine Ausgaben aus.
- Bei Plannerv1 überspringt {"$skip“ :n} nicht, wenn in der vorherigen Phase weniger als n Dokumente zurückgegeben wurden. Plannerv3 und MongoDB überspringen korrekt, unabhängig von der Anzahl der zurückgegebenen Dokumente.
- Wenn eine ausländische Sammlung, auf die in $lookup verwiesen wird, nicht existiert, gibt plannerv1 einen Fehler aus. Planner v3 und MongoDB behandeln die ausländische Sammlung als leere Sammlung und führen $lookup durch.

```
db.coll.aggregate([
    {$lookup: {from: "does_not_exist", localField: "a", foreignField: "a", as:
 "c"}}
])
```

- Nur Planner v1 erlaubt die Verwendung mehrerer $search in einer Pipeline. Planner v2/v3 gibt einen Fehler aus und MongoDB unterstützt ihn nicht.

```
VectorSearch = { "$search": { "vectorSearch": {
   "vector": [0.2, 0.5, 0.8],
   "path": "vectorEmbedding",
   "similarity": "cosine",
   "k": 2,
   "efSearch": 1
   }}}
db.coll.aggregate([VectorSearch, VectorSearch])
```

- Nur Planner v3 funktioniert, wenn die VectorSearch-Stufe nicht die erste Stufe ist. In diesem Fall gibt MongoDB einen Fehler aus und Planner v1 unterstützt die $VectorSearch-Stufe nicht.

```
VectorSearch = { {"$vectorSearch": {
    "queryVector": [0.2, 0.5, 0.8],
    "path": "vectorEmbedding",
```

```
    "similarity": "euclidean",
    "limit": 4,
    "numCandidates": 100} }

db.coll.aggregate([$match:{}, VectorSearch])
```

Abfragen von Geodaten mit Amazon DocumentDB

In diesem Abschnitt wird beschrieben, wie Sie Geodaten mit Amazon DocumentDB abfragen können. Nachdem Sie diesen Abschnitt gelesen haben, können Sie beantworten, wie Geodaten in Amazon DocumentDB gespeichert, abgefragt und indexiert werden.

Themen

- -Übersicht
- Indizierung und Speicherung von Geodaten
- Abfragen von koordinatenbasierten Daten
- Einschränkungen

-Übersicht

Zu den häufigsten Anwendungsfällen für Geospatial gehört die Näherungsanalyse Ihrer Daten. Zum Beispiel „alle Flughäfen im Umkreis von 50 Meilen von Seattle finden“ oder „Finden Sie die nächstgelegenen Restaurants von einem bestimmten Standort aus“. Amazon DocumentDB verwendet die GeoJSON-Spezifikation zur Darstellung von Geodaten. GeoJSON ist eine Open-Source-Spezifikation für die Darstellung JSON-formatting von Formen in einem Koordinatenraum. GeoJSON-Koordinaten erfassen sowohl den Längen- als auch den Breitengrad und stellen Positionen auf einer erdähnlichen Kugel dar.

Indizierung und Speicherung von Geodaten

Amazon DocumentDB verwendet den GeoJSON-Typ „Point“ zum Speichern von Geodaten. Jedes GeoJSON-Dokument (oder Unterdokument) besteht im Allgemeinen aus zwei Feldern:

- type — die dargestellte Form, die Amazon DocumentDB darüber informiert, wie das Feld „Koordinaten“ zu interpretieren ist. Derzeit unterstützt Amazon DocumentDB nur Punkte
- Koordinaten — ein Breiten- und Längengradpaar, das als Objekt in einem Array dargestellt wird — [Längengrad, Breitengrad]

Amazon DocumentDB verwendet auch 2dSphere-Indizes, um Geodaten zu indizieren. Amazon DocumentDB unterstützt Indexierungspunkte. Amazon DocumentDB unterstützt Proximity-Abfragen mit 2dsphere-Indizierung.

Stellen wir uns ein Szenario vor, in dem Sie eine Anwendung für den Lieferservice von Lebensmitteln erstellen. Sie möchten Breiten- und Längengradpaare für Restaurants in Amazon DocumentDB speichern. Erstellen Sie dazu zunächst einen Index für das Geodatenfeld, das das Breiten- und Längengradpaar enthält.

```
use restaurantsdb
db.usarestaurants.createIndex({location:"2dsphere"})
```

Die Ausgabe dieses Befehls würde ungefähr so aussehen:

```
{
 "createdCollectionAutomatically" : true,
 "numIndexesBefore" : 1,
 "numIndexesAfter" : 2,
 "ok" : 1
}
```

Sobald Sie einen Index erstellt haben, können Sie mit dem Einfügen von Daten in Ihre Amazon DocumentDB-Sammlung beginnen.

```
db.usarestaurants.insert({
   "state":"Washington",
   "city":"Seattle",
   "name":"Thai Palace",
   "rating": 4.8,
   "location":{
      "type":"Point",
      "coordinates":[
         -122.3264,
         47.6009
      ]
   }
});

db.usarestaurants.insert({
   "state":"Washington",
   "city":"Seattle",
```

```
   "name":"Noodle House",
   "rating": 4.8,
   "location":{
      "type":"Point",
      "coordinates":[
        -122.3517,
         47.6159
      ]
   }
});

db.usarestaurants.insert({
   "state":"Washington",
   "city":"Seattle",
   "name":"Curry House",
   "rating": 4.8,
   "location":{
      "type":"Point",
      "coordinates":[
         -121.4517,
         47.6229
      ]
   }
});
```

Abfragen von koordinatenbasierten Daten

Amazon DocumentDB unterstützt Näherungs-, Inklusions- und Schnittpunktabfragen von Geodaten. Ein gutes Beispiel für eine Näherungsabfrage ist die Suche nach allen Punkten (allen Flughäfen), die weniger als eine bestimmte Entfernung und mehr als eine Entfernung von einem anderen Punkt (Stadt) entfernt sind. Ein gutes Beispiel für eine Inklusionsabfrage ist die Suche nach allen Punkten (allen Flughäfen), die sich in einem bestimmten Gebiet area/polygon (Bundesstaat New York) befinden. Ein gutes Beispiel für eine Kreuzungsabfrage ist die Suche nach einem Polygon (Bundesstaat), das sich mit einem Punkt (Stadt) überschneidet. Sie können die folgenden Geodatenoperatoren verwenden, um Erkenntnisse aus Ihren Daten zu gewinnen.

- **`$nearSphere`**— `$nearSphere` ist ein Suchoperator, der das Auffinden von Punkten unterstützt, die einem GeoJSON-Punkt am nächsten zum nächsten am weitesten entfernt sind.
- **`$geoNear`**— `$geoNear` ist ein Aggregationsoperator, der die Berechnung der Entfernung von einem GeoJSON-Punkt in Metern unterstützt.

- **`$minDistance`**- `$minDistance` ist ein Suchoperator, der in Verbindung mit `$nearSphere` oder `$geoNear` zum Filtern von Dokumenten verwendet wird, die mindestens den angegebenen Mindestabstand vom Mittelpunkt haben.
- **`$maxDistance`**— `$maxDistance` ist ein Suchoperator, der in Verbindung mit `$nearSphere` oder `$geoNear` zum Filtern von Dokumenten verwendet wird, die sich höchstens in der angegebenen maximalen Entfernung vom Mittelpunkt befinden.
- **`$geoWithin`**— `$geoWithin` ist ein Suchoperator, der die Suche nach Dokumenten mit Geodaten unterstützt, die sich vollständig innerhalb einer bestimmten Form, z. B. eines Polygons, befinden.
- **`$geoIntersects`**- `$geoIntersects` ist ein Suchoperator, der die Suche nach Dokumenten unterstützt, deren Geodaten sich mit einem bestimmten GeoJSON-Objekt überschneiden.

> **Note**
>
> `$geoNear`und `$nearSphere` benötigen einen 2dSphere-Index für das GeoJSON-Feld, das Sie in Ihrer Näherungsabfrage verwenden.

Beispiel 1

In diesem Beispiel erfahren Sie, wie Sie alle Restaurants (Punkte) finden, sortiert nach der nächstgelegenen Entfernung zu einer Adresse (Punkt).

Um eine solche Abfrage durchzuführen, können Sie `$geoNear` die Entfernung einer Reihe von Punkten von einem anderen Punkt berechnen. Sie können das auch hinzufügen`distanceMultiplier`, um die Entfernung in Kilometern zu messen.

```
db.usarestaurants.aggregate([
   {
      "$geoNear":{
         "near":{
            "type":"Point",
            "coordinates":[
               -122.3516,
               47.6156
            ]
         },
         "spherical":true,
         "distanceField":"DistanceKilometers",
         "distanceMultiplier":0.001
```

```
      }
   }
])
```

Mit dem obigen Befehl würden Restaurants nach Entfernung (am nächsten bis am weitesten) vom angegebenen Punkt sortiert zurückgegeben. Die Ausgabe dieses Befehls würde ungefähr so aussehen

```
{ "_id" : ObjectId("611f3da985009a81ad38e74b"), "state" : "Washington", "city" :
 "Seattle", "name" : "Noodle House", "rating" : 4.8, "location" : { "type" : "Point",
 "coordinates" : [ -122.3517, 47.6159 ] }, "DistanceKilometers" : 0.03422834547294996 }
{ "_id" : ObjectId("611f3da185009a81ad38e74a"), "state" : "Washington", "city" :
 "Seattle", "name" : "Thai Palace", "rating" : 4.8, "location" : { "type" : "Point",
 "coordinates" : [ -122.3264, 47.6009 ] }, "DistanceKilometers" : 2.5009390081704277 }
{ "_id" : ObjectId("611f3dae85009a81ad38e74c"), "state" : "Washington", "city" :
 "Seattle", "name" : "Curry House", "rating" : 4.8, "location" : { "type" : "Point",
 "coordinates" : [ -121.4517, 47.6229 ] }, "DistanceKilometers" : 67.52845344856914 }
```

> **Note**
>
> Die num Optionen `limit` und für die `$geoNear` Phase sowie die Standardbeschränkung von 100 Dokumenten sind in Amazon DocumentDB 8.0 (Engine-Patch Version 4.0.5748) und höher veraltet. Die folgenden Beispiele verwenden Amazon DocumentDB 8.0 (Engine-Patch Version 4.0.4448) `limit` und frühere Versionen und num gelten nur für diese.

Verwenden Sie die Option oder, um die Anzahl der Ergebnisse einer Abfrage zu begrenzen. `limit` num

`limit`:

```
db.usarestaurants.aggregate([
   {
      "$geoNear":{
         "near":{
            "type":"Point",
            "coordinates":[
               -122.3516,
               47.6156
            ]
         },
```

```
         "spherical":true,
         "distanceField":"DistanceKilometers",
         "distanceMultiplier":0.001,
         "limit": 10
      }
   }
])
```

num:

```
db.usarestaurants.aggregate([
   {
      "$geoNear":{
         "near":{
            "type":"Point",
            "coordinates":[
               -122.3516,
               47.6156
            ]
         },
         "spherical":true,
         "distanceField":"DistanceKilometers",
         "distanceMultiplier":0.001,
         "num": 10
      }
   }
])
```

Beispiel 2

In diesem Beispiel erfahren Sie, wie Sie alle Restaurants (Punkte) im Umkreis von 2 Kilometern von einer bestimmten Adresse (Punkt) finden. Um eine solche Abfrage durchzuführen, können Sie `$nearSphere` innerhalb eines Mindest `$minDistance` - und Höchstwerts `$maxDistance` von einem GeoJSON-Punkt aus verwenden

```
db.usarestaurants.find({
   "location":{
      "$nearSphere":{
         "$geometry":{
            "type":"Point",
            "coordinates":[
               -122.3516,
```

```
                47.6156
            ]
         },
         "$minDistance":1,
         "$maxDistance":2000
      }
   }
},
{
   "name":1
})
```

Mit dem obigen Befehl würden Restaurants zurückgegeben, die sich in einer maximalen Entfernung von 2 Kilometern vom angegebenen Punkt befinden. Die Ausgabe dieses Befehls würde ungefähr so aussehen

```
{ "_id" : ObjectId("611f3da985009a81ad38e74b"), "name" : "Noodle House" }
```

Einschränkungen

Amazon DocumentDB unterstützt weder die Abfrage noch die Indizierung von Polygonen,, LineString, MultiPoint und. MultiPolygon MultiLineString GeometryCollection

Teilweiser Index

Ein partieller Index indiziert Dokumente in einer Sammlung, die ein bestimmtes Filterkriterium erfüllen. Die Funktion für den teilweisen Index wird in instanzbasierten Amazon DocumentDB 5.0-Clustern unterstützt.

Topics

- Erstellen Sie einen Teilindex
- Unterstützte Operatoren
- Abfrage mit einem Teilindex
- Funktionen des partiellen Indexes
- Einschränkungen des partiellen Indexes

Erstellen Sie einen Teilindex

Um einen Teilindex zu erstellen, verwenden Sie die `createIndex()` Methode mit der `partialFilterExpression` Option. Mit der folgenden Operation wird beispielsweise ein eindeutiger zusammengesetzter Index in der Orders-Auflistung erstellt, der Dokumente indexiert, die ein `OrderID` und das `isDelivered` Feld den Wert true haben:

```
db.orders.createIndex(
  {"category": 1, "CustomerId": 1, "OrderId": 1},
  {"unique": true, "partialFilterExpression":
    {"$and": [
      {"OrderId": {"$exists": true}},
      {"isDelivered": {"$eq": false}}
    ]}
  }
)
```

Unterstützte Operatoren

- $eq
- $exists
- $and (nur auf oberster Ebene)
- $gt/$gte/$lt/$lte (Indexscan wird nur verwendet, wenn der in der Abfrage angegebene Filter exakt mit dem partiellen Filterausdruck übereinstimmt) (siehe Einschränkungen)

Abfrage mit einem Teilindex

Die folgenden Abfragemuster sind mit partiellen Indizes möglich:

- Das Abfrageprädikat entspricht exakt dem Filterausdruck für den partiellen Index:

```
db.orders.find({"$and": [
    {"OrderId": {"$exists": true}},
    {"isDelivered": {"$eq": false}}
  ]}).explain()
```

- Das erwartete Ergebnis des Abfragefilters ist eine logische Teilmenge des Teilfilters:

```
db.orders.find({"$and": [
```

```
    {"OrderId": {"$exists": true}},
    {"isDelivered": {"$eq": false}},
    {"OrderAmount": {"$eq": "5"}}
  ]}).explain()
```

- Ein Unterprädikat der Abfrage kann in Verbindung mit anderen Indizes verwendet werden:

```
db.orders.createIndex({"anotherIndex":1})
db.orders.find({ "$or": [
      {"$and": [
        {"OrderId": {"$exists": true}},
        {"isDelivered": {"$eq": false}}
      ]},
      {"anotherIndex": {"$eq": 5}}
    ]
  }).explain()
```

> **Note**
>
> Ein Abfrageplaner kann sich dafür entscheiden, einen Sammlungsscan anstelle eines Indexscans zu verwenden, wenn dies effizient ist. Dies ist in der Regel bei sehr kleinen Sammlungen oder Abfragen der Fall, bei denen ein großer Teil einer Sammlung zurückgegeben würde.

Funktionen des partiellen Indexes

Listet Teilindizes auf

Listet Teilindizes auf, bei denen die Operation teilweise FilterExpression verwendet wird`getIndex`. Die in ausgegebene `getIndex` Operation listet beispielsweise Teilindizes mit den Feldern key, name und partialFilterExpressions auf:

```
db.orders.getIndexes()
```

Dieses Beispiel gibt die folgende Ausgabe zurück:

```
[
  {
```

```
    "v" : 4,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_",
    "ns" : "ecommerceApp.orders"
  },
  {
    "v" : 4,
    "unique" : true,
    "key" : {
      "category" : 1,
      "" : 1,
      "CustomerId" : 1,
      "OrderId" : 1
    },
    "name" : "category_1_CustID_1_OrderId_1",
    "ns" : "ecommerceApp.orders",
    "partialFilterExpression" : {
      "$and" : [
        {"OrderId": {"$exists": true}},
        {"isDelivered": {"$eq": false}}
      ]
    }
  }
]
```

Mehrere partielle Filterausdrücke für denselben Schlüssel:Reihenfolge

Für dieselben Feldkombinationen (Schlüssel:Reihenfolge) können unterschiedliche Teilindizes erstellt werden. Diese Indizes müssen einen anderen Namen haben.

```
db.orders.createIndex(
  {"OrderId":1},
  {
    name:"firstPartialIndex",
    partialFilterExpression:{"OrderId":{"$exists": true}}
  }
)
```

```
db.orders.createIndex(
  {"OrderId":1},
  {
```

```
    name:"secondPartialIndex",
    partialFilterExpression:{"OrderId":{"$gt": 1000}}
  }
)
```

Führen Sie `getIndexes` den Vorgang aus, um alle Indizes in der Sammlung aufzulisten:

```
db.orders.getIndexes()
```

Diese Beispiele geben die folgende Ausgabe zurück:

```
[
  {
    "v" : 4,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_",
    "ns" : "ecommerceApp.orders"
  },
  {
    "v" : 4,
    "key" : {
      "OrderId" : 1
    },
    "name" : "firstPartialIndex",
    "ns" : "ecommerceApp.orders",
    "partialFilterExpression" : {"OrderId":{"$exists": true}}
  },
  {
    "v" : 4,
    "key" : {
      "OrderId" : 1
    },
    "name" : "secondPartialIndex",
    "ns" : "ecommerceApp.orders",
    "partialFilterExpression" : {"OrderId":{"$gt": 1000}}
  }
]
```

> **Important**
>
> Indexnamen müssen unterschiedlich sein und dürfen nur nach Namen gelöscht werden.

Indizes mit partiellen Eigenschaften und TTL-Eigenschaften

Sie können auch Indizes mit partiellen Eigenschaften und TTL-Eigenschaften erstellen, indem Sie bei der Indexerstellung beide `partialFilterExpression` `expireAfterSeconds` Optionen angeben. Auf diese Weise können Sie besser kontrollieren, welche Dokumente jetzt aus einer Sammlung entfernt werden.

Beispielsweise verfügen Sie möglicherweise über einen TTL-Index, der Dokumente identifiziert, die nach einem bestimmten Zeitraum gelöscht werden sollen. Mit der Option „Teilindex“ können Sie jetzt zusätzliche Bedingungen dafür festlegen, wann Dokumente gelöscht werden sollen:

```
db.orders.createIndex(
    { "OrderTimestamp": 1 },
    {
        expireAfterSeconds: 3600 ,
        partialFilterExpression: { "isDelivered": { $eq: true } }
    }
)
```

Dieses Beispiel gibt die folgende Ausgabe zurück:

```
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1,
        "operationTime" : Timestamp(1234567890, 1)
}
```

Führen Sie den `getIndexes` Vorgang aus, um die in der Sammlung vorhandenen Indizes aufzulisten:

```
db.orders.getIndexes()
[
    {
```

```
        "v" : 4,
        "key" : {
            "_id" : 1
        },
        "name" : "_id_",
        "ns" : "test.orders"
    }
```

Dieses Beispiel gibt die folgende Ausgabe zurück:

```
[
    {
        "v": 4,
        "key": {
            "_id": 1
        },
        "name": "_id_",
        "ns": "ecommerceApp.orders"
    },
    {
        "v": 4,
        "key": {
            "OrderTimestamp": 1
        },
        "name": "OrderTimestamp_1",
        "ns": "ecommerceApp.orders",
        "partialFilterExpression": {
            "isDelivered": {
                "$eq": true
            }
        },
        "expireAfterSeconds": 3600
    }
]
```

Einschränkungen des partiellen Indexes

Für die Funktion „Teilindex“ gelten die folgenden Einschränkungen:

- Ungleichheitsabfragen in Amazon DocumentDB verwenden nur dann einen Teilindex, wenn das Abfragefilterprädikat genau dem entspricht `partialFilterExpression` und denselben Datentyp hat.

> **Note**
>
> `$hint`Kann im obigen Fall sogar nicht verwendet werden, um IXSCAN zu erzwingen.

Im folgenden Beispiel `partialFilterExpression` wird das nur angewendet, `field1` aber nicht`field2`:

```
db.orders.createIndex(
  {"OrderAmount": 1},
  {"partialFilterExpression": { OrderAmount : {"$gt" : 5}}}
)

db.orders.find({OrderAmount : {"$gt" : 5}}) // Will use partial index
db.orders.find({OrderAmount : {"$gt" : 6}}) // Will not use partial index
db.orders.find({OrderAmount : {"$gt" : Decimal128(5.00)}}) // Will not use partial
 index
```

- A `partialFilterExpression` mit Array-Operatoren werden nicht unterstützt. Der folgende Vorgang generiert einen Fehler:

```
db.orders.createIndex(
  {"CustomerId":1},
  {'partialFilterExpression': {'OrderId': {'$eq': [1000, 1001, 1002]}}}
)
```

- Die folgenden Operatoren werden in einem FilterExpression Teilfeld nicht unterstützt:
  - `$all`(Array-Operator)
  - `$mod`(Array-Operator)
  - `$or`
  - `$xor`
  - `$not`
  - `$nor`
- Der Datentyp des Filterausdrucks und des Filters sollten identisch sein.

Textsuche mit Amazon DocumentDB durchführen

Mit der systemeigenen Volltextsuchfunktion von Amazon DocumentDB (Text Index v1) können Sie mithilfe spezieller Textindizes eine Textsuche in großen Textdatensätzen durchführen. Dieser Abschnitt beschreibt die Funktionen der Textindexfunktion und enthält Schritte zur Erstellung und Verwendung von Textindizes in Amazon DocumentDB. Einschränkungen bei der Textsuche sind ebenfalls aufgeführt.

Topics

- Unterstützte Funktionen
- Verwenden Sie den Amazon DocumentDB-Textindex
- Unterschiede zu MongoDB
- Bewährte Verfahren und Richtlinien
- Textindex V2
- Einschränkungen

Unterstützte Funktionen

Die Amazon DocumentDB-Textsuche unterstützt die folgenden MongoDB-API-kompatiblen Funktionen:

- Erstellen Sie Textindizes für ein einzelnes Feld.
- Erstellen Sie zusammengesetzte Textindizes, die mehr als ein Textfeld enthalten.
- Führen Sie Suchen mit einem oder mehreren Wörtern durch.
- Kontrollieren Sie die Suchergebnisse mithilfe von Gewichtungen.
- Sortieren Sie die Suchergebnisse nach Punktzahl.
- Verwenden Sie den Textindex in der Aggregationspipeline.
- Suchen Sie nach einer exakten Phrase.

Verwenden Sie den Amazon DocumentDB-Textindex

Um einen Textindex für ein Feld zu erstellen, das Zeichenkettendaten enthält, geben Sie die Zeichenfolge „Text“ wie unten gezeigt an:

Einzelfeldindex:

```
db.test.createIndex({"comments": "text"})
```

Dieser Index unterstützt Textsuchabfragen im Zeichenfolgenfeld „Kommentare“ in der angegebenen Sammlung.

Erstellen Sie einen zusammengesetzten Textindex für mehr als ein Zeichenfolgenfeld:

```
db.test.createIndex({"comments": "text", "title":"text"})
```

Dieser Index unterstützt Textsuchabfragen in den Zeichenfolgenfeldern „Kommentare“ und „Titel“ in der angegebenen Sammlung. Sie können bis zu 30 Felder angeben, wenn Sie einen Index für zusammengesetzten Text erstellen. Nach der Erstellung fragen Ihre Textsuchabfragen alle indizierten Felder ab.

> **Note**
>
> Für jede Sammlung ist nur ein Textindex zulässig.

Einen Textindex in einer Amazon DocumentDB-Sammlung auflisten

Sie können es in Ihrer Sammlung verwenden`getIndexes()`, um Indizes, einschließlich Textindizes, zu identifizieren und zu beschreiben, wie im folgenden Beispiel gezeigt:

```
rs0:PRIMARY> db.test.getIndexes()
[
   {
      "v" : 4,
      "key" : {
         "_id" : 1
      },
      "name" : "_id_",
      "ns" : "test.test"
   },
   {
      "v" : 1,
      "key" : {
         "_fts" : "text",
```

```
         "_ftsx" : 1
      },
      "name" : "contents_text",
      "ns" : "test.test",
      "default_language" : "english",
      "weights" : {
         "comments" : 1
      },
      "textIndexVersion" : 1
   }
]
```

Sobald Sie einen Index erstellt haben, beginnen Sie mit dem Einfügen von Daten in Ihre Amazon DocumentDB-Sammlung.

```
db.test.insertMany([{"_id": 1, "star_rating": 4, "comments": "apple is red"},
                    {"_id": 2, "star_rating": 5, "comments": "pie is delicious"},
                    {"_id": 3, "star_rating": 3, "comments": "apples, oranges - healthy
 fruit"},
                    {"_id": 4, "star_rating": 2, "comments": "bake the apple pie in the
 oven"},
                    {"_id": 5, "star_rating": 5, "comments": "interesting couch"},
                    {"_id": 6, "star_rating": 5, "comments": "interested in couch for
 sale, year 2022"}])
```

Ausführen von Textsuchabfragen

Führen Sie eine Textsuchabfrage mit einem Wort aus

Sie müssen die `$search` Operatoren `$text` und verwenden, um eine Textsuche durchzuführen. Im folgenden Beispiel werden alle Dokumente zurückgegeben, in denen Ihr textindiziertes Feld die Zeichenfolge „apple“ oder „apple“ in anderen Formaten wie „apples“ enthält:

```
db.test.find({$text: {$search: "apple"}})
```

Ausgabe:

Die Ausgabe dieses Befehls sieht ungefähr so aus:

```
{ "_id" : 1, "star_rating" : 4, "comments" : "apple is red" }
{ "_id" : 3, "star_rating" : 3, "comments" : "apples, oranges - healthy fruit" }
```

```
{ "_id" : 4, "star_rating" : 2, "comments" : "bake the apple pie in the oven" }
```

Führen Sie eine Textsuche mit mehreren Wörtern durch

Sie können auch eine Textsuche mit mehreren Wörtern in Ihren Amazon DocumentDB-Daten durchführen. Der folgende Befehl gibt Dokumente mit einem textindizierten Feld zurück, das „Apfel“ oder „Kuchen“ enthält:

```
db.test.find({$text: {$search: "apple pie"}})
```

Ausgabe:

Die Ausgabe dieses Befehls sieht ungefähr so aus:

```
{ "_id" : 1, "star_rating" : 4, "comments" : "apple is red" }
{ "_id" : 2, "star_rating" : 5, "comments" : "pie is delicious" }
{ "_id" : 3, "star_rating" : 3, "comments" : "apples, oranges - healthy fruit" }
{ "_id" : 4, "star_rating" : 2, "comments" : "bake the apple pie in the oven" }
```

Führen Sie eine Textsuche mit mehreren Wörtern durch

Verwenden Sie für eine Phrasensuche mit mehreren Wörtern dieses Beispiel:

```
db.test.find({$text: {$search: "\"apple pie\""}})
```

Ausgabe:

Der obige Befehl gibt Dokumente mit einem textindizierten Feld zurück, das den exakten Ausdruck „Apfelkuchen“ enthält. Die Ausgabe dieses Befehls sieht ungefähr so aus:

```
{ "_id" : 4, "star_rating" : 2, "comments" : "bake the apple pie in the oven" }
```

Führen Sie eine Textsuche mit Filtern durch

Sie können die Textsuche auch mit anderen Abfrageoperatoren kombinieren, um Ergebnisse anhand zusätzlicher Kriterien zu filtern:

```
db.test.find({$and: [{star_rating: 5}, {$text: {$search: "interest"}}]})
```

Ausgabe:

Der obige Befehl gibt Dokumente mit einem textindizierten Feld zurück, das eine beliebige Form von „Zinsen“ enthält und deren „star_rating“ dem Wert 5 entspricht. Die Ausgabe dieses Befehls sieht ungefähr so aus:

```
{ "_id" : 5, "star_rating" : 5, "comments" : "interesting couch" }
{ "_id" : 6, "star_rating" : 5, "comments" : "interested in couch for sale, year
 2022" }
```

Begrenzen Sie die Anzahl der Dokumente, die bei einer Textsuche zurückgegeben werden

Sie können die Anzahl der zurückgegebenen Dokumente einschränken, indem Sie Folgendes verwenden`limit`:

```
db.test.find({$and: [{star_rating: 5}, {$text: {$search: "couch"}}]}).limit(1)
```

Ausgabe:

Der obige Befehl gibt ein Ergebnis zurück, das den Filter erfüllt:

```
{ "_id" : 5, "star_rating" : 5, "comments" : "interesting couch" }
```

Sortiert die Ergebnisse nach Textnote

Im folgenden Beispiel werden die Ergebnisse der Textsuche nach dem Textwert sortiert:

```
db.test.find({$text: {$search: "apple"}}, {score: {$meta: "textScore"}}).sort({score:
 {$meta: "textScore"}})
```

Ausgabe:

Der obige Befehl gibt Dokumente mit einem textindizierten Feld zurück, das „Apfel“ oder „Apfel“ in seinen anderen Formaten wie „Äpfel“ enthält, und sortiert das Ergebnis danach, wie relevant das Dokument mit dem Suchbegriff zusammenhängt. Die Ausgabe dieses Befehls sieht ungefähr so aus:

```
{ "_id" : 1, "star_rating" : 4, "comments" : "apple is red", "score" :
 0.6079270860936958 }
{ "_id" : 3, "star_rating" : 3, "comments" : "apples, oranges - healthy fruit",
 "score" : 0.6079270860936958 }
```

```
{ "_id" : 4, "star_rating" : 2, "comments" : "bake the apple pie in the oven",
 "score" : 0.6079270860936958 }
```

`$text`und `$search` werden auch für `delete` Befehle `aggregate`count, `findAndModify`update, und unterstützt.

Operatoren für Aggregation

Aggregationspipeline unter Verwendung **`$match`**

```
db.test.aggregate(
   [{ $match: { $text: { $search: "apple pie" } } }]
)
```

Ausgabe:

Der obige Befehl gibt die folgenden Ergebnisse zurück:

```
{ "_id" : 1, "star_rating" : 4, "comments" : "apple is red" }
{ "_id" : 3, "star_rating" : 3, "comments" : "apples, oranges - healthy fruit" }
{ "_id" : 4, "star_rating" : 2, "comments" : "bake the apple pie in the oven" }
{ "_id" : 2, "star_rating" : 5, "comments" : "pie is delicious" }
```

Eine Kombination anderer Aggregationsoperatoren

```
db.test.aggregate(
   [
      { $match: { $text: { $search: "apple pie" } } },
      { $sort: { score: { $meta: "textScore" } } },
      { $project: { score: { $meta: "textScore" } } }
   ]
)
```

Ausgabe:

Der obige Befehl gibt die folgenden Ergebnisse zurück:

```
{ "_id" : 4, "score" : 0.6079270860936958 }
{ "_id" : 1, "score" : 0.3039635430468479 }
{ "_id" : 2, "score" : 0.3039635430468479 }
{ "_id" : 3, "score" : 0.3039635430468479 }
```

Geben Sie mehrere Felder an, wenn Sie einen Textindex erstellen

Sie können bis zu drei Feldern in Ihrem zusammengesetzten Textindex Gewichtungen zuweisen. Die Standardgewichtung, die einem Feld in einem Textindex zugewiesen wird, ist eins (1). Die Gewichtung ist ein optionaler Parameter und muss im Bereich von 1 bis 100000 liegen.

```
db.test.createIndex(
   {
     "firstname": "text",
     "lastname": "text",
     ...
   },
   {
     weights: {
       "firstname": 5,
       "lastname":10,
       ...
     },
     name: "name_text_index"
   }
 )
```

Unterschiede zu MongoDB

Die Textindexfunktion von Amazon DocumentDB verwendet einen invertierten Index mit einem Begriffshäufigkeitsalgorithmus. Textindizes sind standardmäßig spärlich. Aufgrund von Unterschieden in der Analyselogik, Tokenisierungstrennzeichen usw. wird möglicherweise nicht dieselbe Ergebnismenge wie MongoDB für denselben Datensatz oder dieselbe Abfrageform zurückgegeben.

Die folgenden zusätzlichen Unterschiede zwischen dem Amazon DocumentDB-Textindex und MongoDB bestehen:

- Zusammengesetzte Indizes, die keine Textindizes verwenden, werden nicht unterstützt.
- Amazon DocumentDB-Textindizes unterscheiden nicht zwischen Groß- und Kleinschreibung.
- Nur die englische Sprache wird mit dem Textindex unterstützt.
- Die Textindizierung von Array-Feldern (oder Feldern mit mehreren Schlüsseln) wird nicht unterstützt. Beispielsweise schlägt das Erstellen eines Textindexes für „a“ mit dem Dokument {„a“: [„apple“, „pie“]} fehl.
- Die Platzhalter-Textindizierung wird nicht unterstützt.

- Eindeutige Textindizes werden nicht unterstützt.
- Das Ausschließen eines Begriffs wird nicht unterstützt.

Bewährte Verfahren und Richtlinien

- Für eine optimale Leistung bei Textsuchabfragen, bei denen nach Textwerten sortiert wird, erstellen Sie den Textindex, bevor Sie Daten laden.
- Textindizes benötigen zusätzlichen Speicherplatz für eine optimierte interne Kopie der indizierten Daten. Dies hat zusätzliche Auswirkungen auf die Kosten.

Textindex V2

Amazon DocumentDB 8.0 führt eine neue Version des Textindex (V2) ein, die den zugrunde liegenden Textsuchparser ändert, um mehr Kompatibilität mit der MongoDB zu gewährleisten.

Zusätzlich zu den Funktionen, die von den V1-Textindizes bereitgestellt werden, bieten V2-Textindizes auch die folgende Unterstützung:

- Der Planer verschiebt $match-Phasen, wenn möglich, früher in der Pipeline und reduziert so die Anzahl der Dokumente, die in den nachfolgenden Phasen verarbeitet werden.
- Verbesserte Tokenisierung von Sonderzeichen in Textfeldern wie E-Mails, URLs und Dateipfaden. V2 kann einzelne Token innerhalb dieser Zeichenketten analysieren und abgleichen, während V1 dies nicht konnte.

```
rs0:PRIMARY> db.coll.createIndex({ "a": "text" });
rs0:PRIMARY> db.coll.find()
{ "_id" : 1, "a" : "jane.doe_1234@company.com" }
{ "_id" : 2, "a" : "janedoe@company.com" }
{ "_id" : 3, "a" : "/home/user/company/thesis.pdf" }
{ "_id" : 4, "a" : "/home/user/path/jane.pdf" }
{ "_id" : 5, "a" : "http://www.company.com/path" }
{ "_id" : 6, "a" : "https://company.com/path/../home" }

// Sample query
rs0:PRIMARY> db.coll.find({ $text: { $search: "jane" } });

// V1 text index – no results
None
```

```
// V2 text index – matches tokens within emails and file paths
{ "_id" : 1, "a" : "jane.doe_1234@company.com" }
{ "_id" : 4, "a" : "/home/user/path/jane.pdf" }
```

Einschränkungen

Die Textsuche hat in Amazon DocumentDB die folgenden Einschränkungen:

- Textindizes speichern Lexeme und ihre Positionsinformationen. Die Gesamtgröße aller Lexeme und ihrer Positionsinformationen in einem einzigen Dokument ist auf 1 MB begrenzt.

Indizes

Indizes verbessern die Abfrageleistung, da Amazon DocumentDB Dokumente schnell finden kann, ohne jedes Dokument in einer Sammlung scannen zu müssen. Sie können Indizes für Felder erstellen, die Ihre Anwendung häufig abfragt, um Latenz und Kosten zu reduzieren. I/O

Weitere Themen zum Thema Index finden Sie unter:

- Amazon DocumentDB-Indizes verwalten— Indizes erstellen und erstellen
- Arbeiten mit Indizes— Best-Practice-Methoden bei der Indexierung
- Problembehandlung bei Indizes— Behebung von Fehlern bei der Indexerstellung und Überlastung
- Wie analysiere ich die Indexnutzung und identifiziere ungenutzte Indizes?— unbenutzte und fehlende Indizes finden

Themen

- Verwaltung von Indizes
- Indextypen
- Eigenschaften des Indexes

Verwaltung von Indizes

Dieser Abschnitt behandelt Themen zur Indexverwaltung für Amazon DocumentDB.

Themen

- Erstellung eines Indexes
- Verwaltung von Indizes
- Bewährte Methoden für die Arbeit mit Indizes

Erstellung eines Indexes

Für die Erstellung von Indizes in Amazon DocumentDB müssen eine Reihe von Entscheidungen getroffen werden:

- Wie schnell muss es abgeschlossen werden?
- Kann während des Builds auf die Sammlung nicht zugegriffen werden?
- Wie viel Rechenleistung einer Instanz kann dem Build zugewiesen werden?
- Welcher Indextyp sollte erstellt werden?

Dieser Abschnitt hilft Ihnen bei der Beantwortung dieser Fragen und enthält die Befehle und Überwachungsbeispiele, um einen Amazon DocumentDB-Index für Ihre instanzbasierte Clustersammlung zu erstellen.

Richtlinien

Die folgenden Richtlinien beinhalten grundlegende Grenzwerte und Kompromisse bei der Konfiguration bei der Erstellung neuer Indizes:

- Unterstützung für Amazon DocumentDB-Versionen — Während die Single-Worker-Indizierung in allen Amazon DocumentDB-Versionen unterstützt wird, wird die Indexierung mehrerer Worker nur in den Amazon DocumentDB-Versionen 4.0 und 5.0 unterstützt.
- Leistungseinbußen — Eine Erhöhung der Anzahl der Worker im Indexerstellungsprozess erhöht die CPU-Auslastung und die Lese-I/O auf der primären Instance Ihrer Amazon DocumentDB-Datenbank. Die Ressourcen, die für die Erstellung eines neuen Indexes benötigt werden, stehen Ihrem laufenden Workload nicht zur Verfügung.
- Elastische Cluster — Parallele Indizierung wird auf elastischen Amazon DocumentDB-Clustern nicht unterstützt.
- Serverlose Cluster — Für die parallele Indizierung auf serverlosen Clustern ist eine Mindestkapazität von mindestens 32 DCUs erforderlich. Unterhalb dieses Schwellenwerts wird der

`workers` Parameter akzeptiert, aber der Indexaufbau wird mit einem einzigen Worker ausgeführt. Die maximale Anzahl paralleler Worker skaliert mit Ihrer konfigurierten Mindestkapazität: Ungefähr `min_DCU / 8` Worker sind verfügbar. Beispielsweise können Sie bei 32 DCUs bis zu 4 Worker einsetzen, und bei 64 DCUs können Sie bis zu 8 Worker einsetzen.

> **Note**
>
> Nach der Änderung der Mindestkapazitätseinstellung ist ein Cluster-Neustart erforderlich, damit das neue Limit für parallele Worker wirksam wird.

- Maximale Anzahl an Workern (instanzbasiert) — Die maximale Anzahl von Workern, die Sie konfigurieren können, hängt von der Größe Ihrer primären Instance in Ihrem Datenbank-Cluster ab. Dies ist die Hälfte der Gesamtzahl der vCPUs auf der primären Instance Ihres Datenbank-Clusters. Beispielsweise können Sie maximal 32 Worker auf einer db.r6g.16xlarge-Instance mit 64 vCPUs ausführen.

> **Note**
>
> Die parallele Indexerstellung wird auf Instanzen ab 2xlarge unterstützt.

- Mindestanzahl an Arbeitern — Die Mindestanzahl von Arbeitern, die Sie konfigurieren können, ist einer. Die Standardeinstellung für die Indexerstellung auf instanzbasierten Clustern ist zwei Worker. Sie können die Anzahl der Worker jedoch auf einen reduzieren, indem Sie die Option „Worker-Threads“ verwenden. Dadurch wird der Prozess mit einem einzigen Worker ausgeführt.
- Indexkomprimierung — Amazon DocumentDB unterstützt keine Indexkomprimierung. Die Datengrößen für Indizes sind möglicherweise größer als bei Verwendung anderer Optionen.
- Indizierung mehrerer Sammlungen — Die Hälfte der vCPUs auf der primären Instanz Ihres Datenbank-Clusters kann für konfigurierte Worker verwendet werden, die Indexerstellung für mehrere Sammlungen durchführen.
- Indextypen — Eine Liste der unterstützten Indextypen finden Sie Indextypen auf Amazon DocumentDB.

Typen zum Erstellen von Indizes

Die vier Typen von Indexerstellungen sind:

- Vordergrund: Die Erstellung des Vordergrundindexes blockiert alle anderen Datenbankoperationen für die Sammlung, für die der Index erstellt wurde, bis die Indexerstellung abgeschlossen ist. Der Vordergrund-Build von Amazon DocumentDB besteht aus fünf Phasen.
- Vordergrund (einzigartig): Die Erstellung eines Vordergrundindexes für ein einzelnes Dokument (eindeutig) blockiert andere Datenbankoperationen wie reguläre Vordergrund-Builds. Im Gegensatz zum grundlegenden Vordergrund-Build verwendet der einzigartige Build eine zusätzliche Stufe (Sortierung von Schlüsseln 2), um nach doppelten Schlüsseln zu suchen. Der (einzigartige) Vordergrund-Build besteht aus sechs Stufen.
- Hintergrund: Bei der Indexerstellung im Hintergrund können andere Datenbankoperationen im Vordergrund ausgeführt werden, während der Index erstellt wird. Der Amazon DocumentDB-Hintergrund-Build besteht aus acht Phasen.
- Hintergrund (einzigartig): Bei der Erstellung eines (eindeutigen) Hintergrundindexes für ein einzelnes Dokument können andere Datenbankoperationen im Vordergrund ausgeführt werden, während der Index erstellt wird. Im Gegensatz zur grundlegenden Erstellung im Hintergrund verwendet der eindeutige Build eine zusätzliche Phase (Sortierung der Schlüssel 2), um nach doppelten Schlüsseln zu suchen. Der (einzigartige) Hintergrund-Build besteht aus neun Stufen.

Der standardmäßige Indexerstellungstyp ist Vordergrund.

Um einen Index im Hintergrund zu erstellen, fügen Sie beim Erstellen des Indexes den Parameter „background“: true hinzu.

Erste Schritte

Verwenden Sie den Befehl createIndexes, um mit der Indexerstellung für eine Sammlung zu beginnen. Standardmäßig werden mit dem Befehl zwei parallele Worker ausgeführt, wodurch die Geschwindigkeit der Indexerstellung um das Zweifache erhöht wird.

Der folgende Befehlsvorgang veranschaulicht beispielsweise, wie ein Index für das Feld „user_name“ in einem Dokument erstellt und die Geschwindigkeit des Indizierungsprozesses auf vier Worker erhöht wird:

1. Erstellen Sie Indizes im Hintergrund mithilfe von zwei parallelen Workern im Cluster:

```
db.runCommand({"createIndexes":"test","indexes":[{"key": {"user_name":1},
 "name":"username_idx", "background":true }]})
```

2. Um die Geschwindigkeit der Indexerstellung zu optimieren, können Sie die Anzahl der Worker mithilfe der Option „Worker-Threads“ („workers“:<number>) im Befehl db.runCommand createIndexes angeben.

   Erhöhen Sie die Geschwindigkeit des Prozesses auf vier parallele Worker:

```
db.runCommand({"createIndexes":"test","indexes":[{"key": {"user_name":1},
 "name":"username_idx", "workers":4}]})
```

> **Note**
>
> Je höher die Anzahl der Arbeiter ist, desto schneller schreitet die Indexerstellung voran. Je höher jedoch die Anzahl der Worker steigt, desto höher steigt die Last auf den vCPUs und Read I/O Ihrer primären Instance. Stellen Sie sicher, dass Ihr Cluster ausreichend bereitgestellt ist, um die erhöhte Belastung zu bewältigen, ohne andere Workloads zu beeinträchtigen.

Fortschrittsstatus bei der Indizierung

Bei der Indexerstellung werden Sammlungen initialisiert, gescannt, Schlüssel sortiert und schließlich Schlüssel mithilfe eines Indexgenerators eingefügt. Der Prozess umfasst bis zu sechs Stufen, wenn Sie ihn im Vordergrund ausführen, und bis zu neun Stufen, wenn Sie ihn im Hintergrund ausführen. Sie können Statuskennzahlen wie den Prozentsatz der Fertigstellung, die Gesamtzahl der gescannten Speicherblöcke, sortierte Schlüssel und eingefügte Schlüssel schrittweise anzeigen.

Überwachen Sie den Fortschritt des Indizierungsprozesses, indem Sie den Befehl db.CurrentOp () in der Mongo-Shell verwenden. Ein hundertprozentiger Abschluss der letzten Phase zeigt, dass alle Indizes erfolgreich erstellt wurden:

```
db.currentOp({"command.createIndexes": { $exists : true } })
```

> **Note**
>
> Das Anzeigen des Fortschritts der Indizierung wird nur in Amazon DocumentDB 5.0 unterstützt.

Phasen der Indexerstellung

Stage	Vordergrund	Vordergrund (einzigartig)	Hintergrund	Hintergrund (einzigartig)
Initialisieren	1	1	1	1
Index erstellen: initialisieren	2	2	2	2
Gebäudeindex: Sammlung wird gescannt	3	3	3	3
Gebäudeindex: Schlüssel 1 sortieren	4	4	4	4
Gebäudeindex: Schlüssel 2 sortieren		5		5
Gebäudeindex: Schlüssel einfügen	5	6	5	6
validieren: Index scannen			6	7
validieren: Tupel sortieren			7	8

Stage	Vordergrund	Vordergrund (einzigartig)	Hintergrund	Hintergrund (einzigartig)
validieren: Sammlung scannen			8	9

- initialisieren — createIndex bereitet den Index-Builder vor. Diese Phase sollte sehr kurz sein.
- Index erstellen: Initialisieren — Der Indexgenerator bereitet sich auf die Erstellung des Indexes vor. Diese Phase sollte sehr kurz sein.
- Index erstellen: Sammlung scannen — Der Indexgenerator führt einen Sammlungsscan durch, um Indexschlüssel zu sammeln. Die Maßeinheit ist „Blöcke“.

> **Note**
>
> Wenn mehr als ein Worker für die Indexerstellung konfiguriert ist, wird dies in dieser Phase angezeigt. Die Phase „Erfassung scannen“ ist die einzige Phase, in der während der Indexerstellung mehrere Worker verwendet werden. In allen anderen Phasen wird ein einzelner Worker angezeigt.

- Index erstellen: Schlüssel sortieren 1 — Der Indexgenerator sortiert die gesammelten Indexschlüssel. Die Maßeinheit ist „Schlüssel“.
- Building Index: Sorting Keys 2 — Der Index Builder sortiert die gesammelten Indexschlüssel, die toten Tupeln entsprechen. Diese Phase existiert nur für die Erstellung eines eindeutigen Indexes. Die Maßeinheit ist „Schlüssel“.
- Index erstellen: Schlüssel einfügen — Der Indexgenerator fügt Indexschlüssel in den neuen Index ein. Die Maßeinheit ist „Schlüssel“.
- validieren: Index scannen — createIndex durchsucht den Index, um Schlüssel zu finden, die überprüft werden müssen. Die Maßeinheit ist „Blöcke“.
- Validieren: Sortieren von Tupeln — CreateIndex sortiert die Ausgabe der Index-Scan-Phase.
- Validieren: Sammlung wird gescannt — CreateIndex durchsucht die Sammlung, um die Indexschlüssel zu überprüfen, die in den beiden vorherigen Phasen gefunden wurden. Die Maßeinheit ist „Blöcke“.

Beispiel für die Ausgabe der Indexerstellung

Im folgenden Ausgabebeispiel (Erstellung des Vordergrundindexes) wird der Status der Indexerstellung angezeigt. Das Feld „msg“ fasst den Fortschritt des Builds zusammen, indem es die Phase und den Prozentsatz der Fertigstellung des Builds angibt. Das Feld „Workers“ gibt die Anzahl der Worker an, die in dieser Phase der Indexerstellung verwendet wurden. Im Feld „Fortschritt“ werden die tatsächlichen Zahlen angezeigt, anhand derer der Prozentsatz der Fertigstellung berechnet wurde.

> **Note**
>
> Die Felder IndexBuildName „current“, „msg“ und „progress“ werden in Amazon DocumentDB Version 4.0 nicht unterstützt.

```
{
    "inprog" : [{
    …
        "command": {
            "createIndexes": "test",
            "indexes": [{
                "v": 2,
                "key": {
                    "user_name": 1
                },
                "name": "user_name_1"
            }],
            "lsid": {
                "id": UUID("094d0fba-8f41-4373-82c3-7c4c7b5ff13b")
            },
            "$db": "test"
        },
        "currentIndexBuildName": user_name_1,
        "msg": "Index Build: building index number_1, stage 6/6 building index:
 656860/1003520 (keys) 65%",
        "workers": 1,
        "progress": {
            "done": 656861,
            "total": 1003520
        },
    …
```

```
    ],

    "ok" : 1
}
```

Ressourcen zur Fehlerbehebung

In den folgenden Ressourcen finden Sie Informationen zur Behebung von Problemen bei der Indexerstellung:

- Die Indexerstellung schlägt fehl
- Latenzprobleme und Fehlschläge bei der Indexerstellung im Hintergrund

Verwaltung von Indizes

Indexblähung

Amazon DocumentDB verwendet Multi-Version Concurrency Control (MVCC), um gleichzeitige Transaktionen zu verwalten. Wenn Dokumente gelöscht oder aktualisiert werden, verbleiben ihre früheren Versionen als „tote“ Versionen in Sammlungen und Indizes. Der Garbage Collection-Prozess gibt automatisch Speicherplatz aus diesen toten Versionen für zukünftige Operationen zurück.

Eine Aufblähung des Indexes tritt auf, wenn die Indizes einer Sammlung aufgrund der Anhäufung von toten oder veralteten Indexeinträgen oder einer Fragmentierung innerhalb der Seiten größer werden. Der angegebene Prozentsatz steht für die Menge an Indexplatz, die von zukünftigen Indexeinträgen genutzt werden kann. Diese Überlastung verbraucht Speicherplatz sowohl im Puffercache als auch im Speicher. Wenn Sie das Aufblähen entfernen möchten, müssen Sie die Indizes neu erstellen.

Beispiel: Führen Sie den folgenden Befehl aus, um ungenutzten Speicherplatz für Ihren Index zu ermitteln:

```
db.coll.aggregate({$indexStats:{}});
```

Das Ergebnis sieht ungefähr so aus:

```
{
```

```
    "name" : "_id_",
    "key" : {
        "_id" : 1
    },
    "host" : "devbox-test.localhost.a2z.com:27317",
    "size" : NumberLong(827392),
    "accesses" : {
        "ops" : NumberLong(40000),
        "docsRead" : NumberLong(46049),
        "since" : ISODate("2025-04-03T21:44:51.251Z")
    },
    "cacheStats" : {
        "blksRead" : NumberLong(264),
        "blksHit" : NumberLong(140190),
        "hitRatio" : 99.8121
    },
    "unusedStorageSize" : {
        "unusedSizeBytes" : 409600,
        "unusedSizePercent" : 49.51
    }
}
```

Sie können Indizes ohne Ausfallzeiten neu erstellen, indem Sie den Befehl Reindex verwenden, der einen Scan der gesamten Sammlung erfordert. Siehe Indexverwaltung mit Reindex.

Indexverwaltung mit Reindex

Reindex ist ein Befehl, mit dem ein Index neu erstellt wird. Er wird normalerweise verwendet, wenn ein Index beschädigt oder ineffizient geworden ist. Im Laufe der Zeit können Indizes aufgrund vieler Aktualisierungen, Einfügungen oder Löschungen ungenutzten Speicherplatz ansammeln, was zu Leistungseinbußen führt. Die Neuindizierung hilft dabei, diesen ungenutzten Speicherplatz zu entfernen und die Effizienz des Indexes wiederherzustellen.

Richtlinien zur Neuindizierung

- Reindex wird nur auf Amazon DocumentDB 5.0 und höher unterstützt
- Reindex wird immer im Hintergrund ausgeführt.
- Amazon DocumentDB unterstützt die Neuindizierung eines einzelnen Indexes im Hintergrund, sodass mehrere Worker möglich sind. Der alte Index kann für Abfragen verwendet werden, wenn der Reindex-Prozess läuft.

- Amazon DocumentDB unterstützt die Indexierung von Fortschrittsberichten über CurrentOp. Sie können die Phasen der Indexerstellung sehen, die den Phasen der Indexerstellung bei der Indexerstellung ähneln. Der einzige Unterschied besteht darin, dass Reindex immer acht Stufen hat, unabhängig davon, ob es eindeutig ist oder nicht. Es gibt keine Stufe „Index erstellen: Schlüssel sortieren 2“.
- Reindex kann gleichzeitig mit allen Befehlen ausgeführt werden, mit Ausnahme von Index-Befehlen für dieselbe Sammlung: createIndexes, dropIndexes, CollMod und RenameCollection.
- Reindex wird derzeit nicht für Text-, Geodaten- und Vektorindizes unterstützt.
- Die Reindex-Unterstützung für Teilindizes ist in der Engine-Version 8.0.2 und höher verfügbar.

ReIndex erstellen

Verwenden Sie den folgenden Befehl, um Ihren Index neu zu erstellen:

```
db.runCommand({ reIndex: "collection-name", index: "index-name"})
```

Optional können Sie auch die Anzahl der Worker steuern, die dem Neuerstellungsprozess zugewiesen sind:

```
db.runCommand({ reIndex: "collection-name", index: "index-name", workers: number })
```

Spezifische Informationen zur Verwaltung von Indizes mit Java finden Sie unterIndexverwaltung in Amazon DocumentDB mit Java.

Bewährte Methoden für die Arbeit mit Indizes

Indizes erstellen

Wenn Sie Daten in Amazon DocumentDB importieren, sollten Sie Ihre Indizes erstellen, bevor Sie große Datensätze importieren. Sie können das Amazon DocumentDB Index Tool verwenden, um Indizes aus einer laufenden MongoDB-Instance oder einem Mongodump-Verzeichnis zu extrahieren und diese Indizes in einem Amazon DocumentDB-Cluster zu erstellen. Weitere Hinweise zu Migrationen finden Sie unter Migration und Aktualisierung.

Index-Selektivität

Beschränken Sie die Indexerstellung auf Felder, bei denen die Anzahl der doppelten Werte weniger als 1% der Gesamtzahl der Dokumente in der Sammlung beträgt. Wenn Ihre Sammlung beispielsweise 100.000 Dokumente enthält, erstellen Sie Indizes nur für Felder, in denen derselbe Wert 1.000 Mal oder weniger vorkommt.

Wenn Sie einen Index mit einer hohen Anzahl von Einzelwerten (d. h. einer hohen Kardinalität) wählen, wird sichergestellt, dass Filtervorgänge eine kleine Anzahl von Dokumenten zurückgeben, wodurch eine gute Leistung bei Indexscans erzielt wird. Ein Beispiel für einen Index hoher Kardinalität ist ein einzigartiger Index, der garantiert, dass Gleichheitsprädikate höchstens ein einziges Dokument zurückgeben. Beispiele für niedrige Kardinalität sind ein Index über ein boolesches Feld und ein Index über einen Wochentag. Aufgrund ihrer schlechten Leistung ist es unwahrscheinlich, dass Indizes mit niedriger Kardinalität vom Abfrageoptimierer der Datenbank ausgewählt werden. Gleichzeitig verbrauchen Indizes mit niedriger Kardinalität weiterhin Ressourcen wie Speicherplatz und. I/Os Als Faustregel gilt, dass Sie Indizes für Felder verwenden sollten, bei denen die typische Wertehäufigkeit 1 % der Gesamtsammlungsgröße oder weniger beträgt.

Darüber hinaus wird empfohlen, nur Indizes für Felder zu erstellen, die häufig als Filter verwendet werden, und regelmäßig nach nicht verwendeten Indizes zu suchen. Weitere Informationen finden Sie unter the section called “Analysieren Sie die Indexnutzung und identifizieren Sie ungenutzte Indizes”.

Auswirkung von Indizes auf das Schreiben von Daten

Zwar können Indizes die Abfrageleistung verbessern, da nicht jedes Dokument in einer Sammlung gescannt werden muss, diese Verbesserung erfordert jedoch einen Kompromiss. Für jeden Index einer Sammlung muss die Datenbank jedes Mal, wenn ein Dokument eingefügt, aktualisiert oder gelöscht wird, die Sammlung aktualisieren und die Felder in jeden der Indizes für die Sammlung schreiben. Wenn eine Sammlung beispielsweise über neun Indizes verfügt, muss die Datenbank zehn Schreibvorgänge durchführen, bevor die Operation dem Kunden bestätigt wird. Somit führt jeder zusätzliche Index zu einer zusätzlichen Schreiblatenz und einer Erhöhung des insgesamt genutzten Speichers. I/Os

Cluster-Instances müssen über eine angemessene Größe verfügen, um den gesamten Arbeitsspeicher zu erhalten. Dadurch entfällt die Notwendigkeit, ständig Indexseiten vom Speichervolumen zu lesen, was sich negativ auf die Leistung auswirkt und höhere I/O Kosten verursacht. Weitere Informationen finden Sie unter Dimensionierung der Instanz.

Um eine optimale Leistung zu erzielen, sollten Sie die Anzahl der Indizes in Ihren Sammlungen minimieren und nur die Indizes hinzufügen, die zur Verbesserung der Leistung bei häufigen Abfragen

erforderlich sind. Bei variierenden Workloads gilt als gute Richtschnur, die Anzahl der Indizes pro Sammlung auf fünf oder weniger zu beschränken.

Identifizierung fehlender Indizes

Es hat sich bewährt, fehlende Indizes regelmäßig zu identifizieren. Weitere Informationen finden Sie unter the section called “Identifizieren Sie fehlende Indizes”.

Identifizieren ungenutzter Indizes

Es hat sich bewährt, ungenutzte Indizes regelmäßig zu identifizieren und zu entfernen. Weitere Informationen finden Sie unter the section called “Analysieren Sie die Indexnutzung und identifizieren Sie ungenutzte Indizes”.

Indextypen

Themen

- Indizes aus einzelnen Feldern
- Zusammengesetzte Indizes
- Multi-key Indizes
- Textindizes
- Geodatenindizes
- Vektor-Indizes

Indizes aus einzelnen Feldern

Einzelfeldindizes speichern Informationen aus jedem einzelnen Feld in einer Sammlung von Dokumenten. Standardmäßig haben alle Sammlungen einen Index für das `_id` Feld. Sie können zusätzliche Indizes hinzufügen, um wichtige Abfragen und Operationen zu beschleunigen.

Einzelfeldindizes sind in folgenden Fällen von Vorteil:

- Ihre Anwendung fragt Dokumente häufig auf der Grundlage der Werte eines bestimmten Felds ab oder filtert sie.
- Sie müssen Dokumente effizient nach einem bestimmten Feld sortieren.
- Sie möchten sicherstellen, dass ein bestimmtes Feld für alle Dokumente in einer Sammlung eindeutig ist, indem Sie einen eindeutigen Index erstellen.

Unterstützte Indexeigenschaften

Option	3.6	4,0	5.0	8.0	Elastischer Cluster
name	Ja	Ja	Ja	Ja	Ja
einzigartig	Ja	Ja	Ja	Ja	Ja
spärlich *	Ja	Ja	Ja	Ja	Ja
teilweise * FilterExpression	Nein	Nein	Ja	Ja	Nein
ablaufen AfterSeconds	Ja	Ja	Ja	Ja	Ja

* Die `partialFilterExpression` Optionen `sparse` und können nicht zusammen in derselben Indexdefinition verwendet werden. Wenn Sie versuchen, einen Index mit diesen Optionen zu erstellen, schlägt dies mit dem folgenden Fehler fehl:

```
Error in specification: cannot mix partialFilterExpression and sparse options
```

Einen Index für ein einzelnes Feld erstellen

Verwenden Sie die `createIndex()` Methode, um einen einzelnen Feldindex zu erstellen. Die Syntax lautet:`db.collection.createIndex(<key>, <options>)`.

Der `key` Parameter ist ein JSON-Dokument, das das Feld und die Index-Sortierreihenfolge angibt:

```
{
  "<field>": <1 (ascending)|-1 (descending)>
}
```

Der `options` Parameter ist ein JSON-Dokument, das die Optionen für den Index angibt:

```
{
  "name": "<name>",
  "unique": <true | false>,
```

```
  "sparse": <true | false>,
  "partialFilterExpression": <filter expression>,
  "expireAfterSeconds": <seconds before expiry>
}
```

Beispiele für die Erstellung von Einzelfeldindizes finden Sie unter Indexeigenschaften.

Zusammengesetzte Indizes

Zusammengesetzte Indizes speichern Informationen aus zwei oder mehr Feldern in einer Sammlung von Dokumenten, sodass Abfragen im ersten Feld oder in beliebigen Präfixfeldern möglich sind. Diese Indizes optimieren die Leistung von Abfragen, die nach mehreren Feldern gleichzeitig filtern oder Filter mit Sortiervorgängen kombinieren. Sie eignen sich auch für Abfragen mit einer Bedingung in den indizierten Feldern ganz links. Die Datenbank nutzt diese Indexeinträge, um passende Dokumente effizient zu finden, ohne vollständige Sammlungsscans durchführen zu müssen.

Zusammengesetzte Feldindizes sind in folgenden Fällen von Vorteil:

- Sie müssen nach mehreren Feldern gleichzeitig filtern.
- Sie müssen das Filtern mit Sortiervorgängen kombinieren.

Unterstützte Indexeigenschaften

Option	3.6	4,0	5.0	8.0	Elastischer Cluster
name	Ja	Ja	Ja	Ja	Ja
einzigartig	Ja	Ja	Ja	Ja	Ja
spärlich *	Ja	Ja	Ja	Ja	Ja
teilweise * FilterExpression	Nein	Nein	Ja	Ja	Nein

* Die `partialFilterExpression` Optionen `sparse` und können nicht zusammen in derselben Indexdefinition verwendet werden. Wenn Sie versuchen, einen Index mit diesen Optionen zu erstellen, schlägt dies mit dem folgenden Fehler fehl:

```
Error in specification: cannot mix partialFilterExpression and sparse options
```

Einen zusammengesetzten Index erstellen

Verwenden Sie die `createIndex()` Methode, um einen zusammengesetzten Index zu erstellen. Die Syntax der Methode lautet: `db.collection.createIndex(<keys>, <options>)`

Der `keys` Parameter ist ein JSON-Dokument, das die Felder und die Index-Sortierreihenfolge angibt:

```
{
  "<field 1>": <1 (ascending)|-1 (descending)>,
  "<field 2>": <1 (ascending)|-1 (descending)>,
  ...
}
```

Beachten Sie, dass nur ein Feld ein Array in einem zusammengesetzten Index sein kann. Wenn Sie versuchen, einen zusammengesetzten Index für zwei oder mehr Array-Felder zu erstellen, schlägt dies mit dem folgenden Fehler fehl:

```
multiple fields of compound index cannot be arrays
```

Der `options` Parameter ist ein JSON-Dokument, das die Optionen für den Index angibt:

```
{
  "name": "<name>",
  "unique": <true | false>,
  "sparse": <true | false>,
  "partialFilterExpression": <filter expression>
}
```

Beispiele für die Erstellung zusammengesetzter Indizes finden Sie unter Indexeigenschaften.

Multi-key Indizes

Für Felder, die einen Array-Wert haben, ermöglicht Ihnen ein Index mit mehreren Schlüsseln, einen Indexschlüssel für jedes Element im Array zu erstellen. Durch die Indizierung eines Arrays wird ein Indexeintrag für jedes Element des Arrays erstellt.

Multi-key Indizes sind nützlich, wenn Ihre Anwendung häufig Dokumente anhand von Werten in Arrays abfragt oder filtert.

Unterstützte Indexeigenschaften

Option	3.6	4,0	5.0	8.0	Elastischer Cluster
name	Ja	Ja	Ja	Ja	Ja
einzigartig	Ja	Ja	Ja	Ja	Ja
spärlich *	Ja	Ja	Ja	Ja	Ja
teilweise * FilterExpression	Nein	Nein	Ja	Ja	Nein
ablaufen AfterSeconds	Ja	Ja	Ja	Ja	Ja

* Die `partialFilterExpression` Optionen `sparse` und können nicht zusammen in derselben Indexdefinition verwendet werden. Wenn Sie versuchen, einen Index mit diesen Optionen zu erstellen, schlägt dies mit dem folgenden Fehler fehl:

```
Error in specification: cannot mix partialFilterExpression and sparse options
```

Einen Index mit mehreren Schlüsseln erstellen

Verwenden Sie die `createIndex()` Methode, um einen Index mit mehreren Schlüsseln zu erstellen. Die Syntax der Methode lautet: `db.collection.createIndex(<key>, <options>)`

Der `key` Parameter ist ein JSON-Dokument, das das Feld und die Index-Sortierreihenfolge angibt:

```
{
  "<field>": <1 (ascending)|-1 (descending)>
}
```

Der `options` Parameter ist ein JSON-Dokument, das die Optionen für den Index angibt:

```
{
  "name": "<name>",
```

```
  "unique": <true | false>,
  "sparse": <true | false>,
  "partialFilterExpression": <filter expression>,
  "expireAfterSeconds": <seconds before expiry>
}
```

Im folgenden Beispiel wird ein Index mit mehreren Schlüsseln für das `categories` Feld erstellt, das in aufsteigender Reihenfolge mit dem Namen sortiert ist: `book_categories`

```
db.collection.createIndex(
  {
    "categories": 1
  },
  {
    "name": "book_categories"
  }
)
```

Beispiele für das Erstellen von Indizes mit mehreren Schlüsseln finden Sie unter Indexeigenschaften.

Textindizes

Textindizes sind nützlich bei der Suche nach Wörtern oder Ausdrücken in Textfeldern innerhalb von Dokumenten, sodass Sie Volltextsuchfunktionen ausführen können. Sie können Einzelfeld-Textindizes oder zusammengesetzte Textindizes mit mehr als einem Textfeld erstellen. Sie können jedoch nur einen Textindex pro Sammlung haben. Weitere Informationen finden Sie unter Textsuche mit Amazon DocumentDB durchführen.

Unterstützte Indexeigenschaften

Option	3.6	4,0	5.0	8.0	Elastischer Cluster
name	Nein	Nein	Ja	Ja	Nein

Einen Textindex erstellen

Verwenden Sie die `createIndex()` Methode, um einen Textindex zu erstellen. Die Syntax der Methode lautet: `db.collection.createIndex(<keys>, <options>)`

Der `keys` Parameter ist ein JSON-Dokument, das das Feld (die Felder) und den Textindextyp angibt:

```
{
  "<field 1>": "text",
  "<field 2>": "text"
}
```

Der `options` Parameter ist ein JSON-Dokument, das den Namen des Indexes angibt:

```
{
  "name": "<name>"
}
```

Beispiele für das Erstellen von Textindizes finden Sie unter Indexeigenschaften.

Geodatenindizes

Geodatenindizes sind spezielle Indextypen, mit denen Geodaten, die in einer Sammlung von Dokumenten gespeichert sind, effizient abgefragt und verwaltet werden können. Amazon DocumentDB unterstützt 2dSphere-Indizes, die speziell für die Verarbeitung von Geodaten auf einer Kugel (wie der Erde) entwickelt wurden. Dies ermöglicht genaue Berechnungen und Abfragen auf der Grundlage der sphärischen Geometrie.

Geodatenindizes sind nützlich, wenn Ihre Anwendungen standortbezogene Abfragen ausführen müssen, wie z. B.:

- Auffinden von Sehenswürdigkeiten in der Nähe,
- festzustellen, ob ein Standort in ein bestimmtes Gebiet fällt
- Berechnung von Entfernungen zwischen Orten

Unterstützte Indexeigenschaften

Option	3.6	4,0	5.0	8.0	Elastischer Cluster
name	Ja	Ja	Ja	Ja	Ja

Erstellen eines räumlichen Indexes

Verwenden Sie die `createIndex()` Methode, um einen Geodatenindex zu erstellen. Die Syntax der Methode lautet: `db.collection.createIndex(<key>, <options>)`

Der `key` Parameter ist ein JSON-Dokument, das das Feld und den 2dsphere-Indextyp angibt:

```
{
  "<field>": "2dsphere"
}
```

Der `options` Parameter ist ein JSON-Dokument, das die Optionen für den Index angibt:

```
{
  "name": "<name>"
}
```

Beispiele für die Erstellung von Geoindizes finden Sie unter Indexeigenschaften.

Vektor-Indizes

Vektorindizes sind spezielle Indextypen, mit denen Vektordaten, die in einer Sammlung von Dokumenten gespeichert sind, effizient abgefragt und verwaltet werden können. Amazon DocumentDB unterstützt die Indizes Hierarchical Navigable Small World (HNSW) und Inverted File with Flat Compression (IVFflat).

Weitere Informationen finden Sie unter Vektorsuche für Amazon DocumentDB.

Vektorindizes sind für maschinelles Lernen und generative KI-Anwendungsfälle von Vorteil, wie z. B.:

- Semantische Suche
- Produktempfehlung
- Personalisierung
- Chatbots
- Aufdeckung von Betrug
- Erkennung von Anomalien

Unterstützte Indexeigenschaften

Option	3.6	4,0	5.0	8.0	Elastischer Cluster
name	Nein	Nein	Ja	Ja	Nein

Erstellen eines Vektorindexes

Verwenden Sie den Befehl createIndex mit der `runCommand()` Methode, um einen Vektorindex zu erstellen. Die Syntax lautet wie folgt:

```
db.runCommand({
  "createIndexes": "<collection>",
  "indexes": [{
    "key": {
      "<field>": "vector"
    },
    "name": "<name>",
    "vectorOptions": {
      "type": "<hnsw> | <ivfflat>",
      "dimensions": <number of dimensions>,
      "similarity": "<euclidean>|<cosine>|<dotProduct>",
      "lists": <number_of_lists> [applicable for IVFFlat],
      "m": <max number of connections> [applicable for HNSW],
      "efConstruction": <size of the dynamic list for index build> [applicable for
 HNSW]
    }
  }]
})
```

Der Schlüsselparameter ist ein JSON-Dokument, das den Feld- und Vektorindextyp angibt:

```
{
  "<field>": "vector"
}
```

Beispiele für die Erstellung von Vektorindizes finden Sie unter Indexeigenschaften.

Eigenschaften des Indexes

Indexeigenschaften definieren das Verhalten und die Eigenschaften von Indizes in Amazon DocumentDB. Wenn Sie diese Eigenschaften kennen, können Sie effiziente Indizes entwerfen, die den spezifischen Anforderungen Ihrer Anwendung entsprechen.

Topics

- Indexeigenschaft: Name
- Index-Eigenschaft: einzigartig
- Indexeigenschaft: spärlich
- Index-Eigenschaft: teilweise FilterExpression
- Indexeigenschaft: ablaufen AfterSeconds
- Indexeigenschaft: versteckt

Indexeigenschaft: Name

Unterstützte Indextypen

Option	3.6	4,0	5.0	8.0	Elastischer Cluster
einzelnes Feld	Ja	Ja	Ja	Ja	Ja
compound	Ja	Ja	Ja	Ja	Ja
mehrere Schlüssel	Ja	Ja	Ja	Ja	Ja
text	Nein	Nein	Ja	Ja	Nein
Geodaten	Ja	Ja	Ja	Ja	Ja
Vektor	Nein	Nein	Ja	Ja	Nein

Verwenden Sie die Option Name, um einen optionalen Namen für den Index anzugeben.

In allen Beispielen wird das folgende Beispieldokument verwendet:

```
{
  "productId": "PROD133726",
  "sku": "SKU24224",
  "name": "Basic Printer",
  "manufacturer": "The Manufacturer",
  "tags": [ "printer", "basic", "electronics", "business" ],
  "barcodes": [ "542364671", "886330670", "437445606" ],
  "reviews": [
    {
      "review_date": ISODate('2024-01-19T21:37:10.585Z'),
      ...
    }
  ],
  "material": "Polycarbonate",
  "color": "Space Gray",
  "supplier": {
    "supplierId": "SUP4",
    "location": {
      "type": "Point",
      "coordinates": [ -71.0589, 42.3601 ]
    }
  },
  "productEmbedding": [
    -0.019320633663838058,
    0.019672111388113596
  ],
  "lastUpdated": ISODate('2025-10-20T21:37:10.585Z')
}
```

Einzelnes Feld

```
db.collection.createIndex(
  {
    "productId": 1
  },
  {
    "name": "single_field_index"
  }
)
```

Zusammengesetzt

```
db.collection.createIndex(
  {
    "productId": 1,
    "manufacturer": 1
  },
  {
    "name": "compound_index"
  }
)
```

Multi-key

```
db.collection.createIndex(
  {
    "tags": 1
  },
  {
    "name": "multikey_index"
  }
)
```

Text

```
db.collection.createIndex(
  {
    "name": "text"
  },
  {
    "name": "text_index"
  }
)
```

Geodaten

```
db.collection.createIndex(
  {
    "supplier.location": "2dsphere"
  },
  {
```

```
    "name": "geospatial_index"
  }
)
```

Vektor

```
db.runCommand({
  "createIndexes": "collection",
  "indexes": [{
    "key": {
      "productEmbedding": "vector"
    },
    "name": "hnsw_index",
    "vectorOptions": {
      "type": "hnsw",
      "dimensions": 2,
      "similarity": "euclidean",
      "m": 16,
      "efConstruction": 64
    }
  }]
})
```

Index-Eigenschaft: einzigartig

Unterstützte Indextypen

Option	3.6	4,0	5.0	8.0	Elastischer Cluster
einzelnes Feld	Ja	Ja	Ja	Ja	Ja
compound	Ja	Ja	Ja	Ja	Ja
mehrere Schlüssel	Ja	Ja	Ja	Ja	Ja

Verwenden Sie die Option „Eindeutig“, um sicherzustellen, dass die Felder in allen Dokumenten einer Sammlung eindeutig sind.

Beispiele

Die folgenden Beispiele zeigen, wie eindeutige Indizes für das folgende Beispieldokument erstellt werden:

```
{
  "productId": "PROD133726",
  "sku": "SKU24224",
  "name": "Basic Printer",
  "manufacturer": "The Manufacturer",
  "tags": [ "printer", "basic", "electronics", "business" ],
  "barcodes": [ "542364671", "886330670", "437445606" ],
  "reviews": [
    {
      "review_date": ISODate('2024-01-19T21:37:10.585Z'),
      ...
    }
  ],
  "material": "Polycarbonate",
  "color": "Space Gray",
  "supplier": {
    "supplierId": "SUP4",
    "location": {
      "type": "Point",
      "coordinates": [ -71.0589, 42.3601 ]
    }
  },
  "productEmbedding": [
    -0.019320633663838058,
    0.019672111388113596
  ],
  "lastUpdated": ISODate('2025-10-20T21:37:10.585Z')
}
```

einzelnes Feld

Erstellen Sie einen eindeutigen Einzelfeldindex für ProductID, um sicherzustellen, dass dieselbe ProductID nicht in mehr als einem Dokument vorhanden ist:

```
db.collection.createIndex(
  {
    "productId": 1
  },
```

```
  {
    name: "productId_unique",
    unique: true
  }
)
```

Zusammengesetzt

Erstellen Sie einen eindeutigen zusammengesetzten Index für Artikelnummer und Hersteller, um sicherzustellen, dass dieselbe Kombination aus SKU und Hersteller nicht in mehr als einem Dokument vorkommt:

```
db.collection.createIndex(
  {
    "sku": 1,
    "manufacturer": 1
  },
  {
    name: "sku_and_manufacturer_unique",
    unique: true
  }
)
```

mehrere Schlüssel

Erstellen Sie einen eindeutigen Index mit mehreren Schlüsseln für Barcodes, um sicherzustellen, dass ein Wert im Barcode-Array nicht in mehr als einem Dokument vorkommt:

```
db.collection.createIndex(
  {
    "barcodes": 1
  },
  {
    name: "barcodes_unique",
    unique: true
  }
)
```

Durch die Indizierung eines Arrays wird ein Indexeintrag für jedes Element des Arrays erstellt. Wenn ein Array beispielsweise 50 Elemente hat, hat es 50 Indexeinträge. Aus diesem Grund erzwingen eindeutige Indizes mit mehreren Schlüsseln die Eindeutigkeit aller einzelnen Elemente.

Beispielsweise verletzen die folgenden Dokumente die Eindeutigkeitsbeschränkung für den Index des Werte-Array-Felds:

```
{ "values": [ 1, 2, 3] }
{ "values": [ 3, 2 ] }   --> 3 and 2 already exist
{ "values": [ 1 ] }      --> 1 already exists
```

Beachten Sie das folgende Verhalten bei eindeutigen Indizes:

1. Wenn Sie einen eindeutigen Index für vorhandene Daten erstellen, bei denen zwei (oder mehr) Dokumente dieselben Werte für die indizierten Felder haben, schlägt die Indexerstellung mit dem folgenden Fehler fehl: `could not create unique index: <collection> index: <index name>`
2. Wenn Sie ein Dokument einfügen, bei dem der Wert des indizierten Felds dem Wert dieses Felds in einem anderen Dokument entspricht, schlägt das Einfügen fehl und es wird der folgende Fehler angezeigt: `E11000 duplicate key error collection: <collection> index: <index name>`
3. Wenn Sie ein vorhandenes Dokument aktualisieren, sodass der neue Wert des indizierten Felds dem Wert dieses Felds in einem anderen Dokument entspricht, schlägt die Aktualisierung mit dem folgenden Fehler fehl: `E11000 duplicate key error collection: <collection> index: <index name>`
4. Wenn das indizierte Feld in einem Dokument fehlt, wird der Wert als Null behandelt. Indexerstellungen, Einfügungen und Aktualisierungen schlagen wie oben beschrieben fehl, wenn das indizierte Feld in zwei (oder mehr) Dokumenten fehlt.

Indexeigenschaft: spärlich

Unterstützte Indextypen

Option	3.6	4,0	5.0	8.0	Elastischer Cluster
einzelnes Feld	Ja	Ja	Ja	Ja	Ja
compound	Ja	Ja	Ja	Ja	Ja

Option	3.6	4,0	5.0	8.0	Elastischer Cluster
mehrere Schlüssel	Ja	Ja	Ja	Ja	Ja

Verwenden Sie die Option Sparse, um die Indizierung von Dokumenten zu überspringen, denen die indizierten Felder fehlen. Dadurch wird die Indexgröße reduziert und Speicherplatz gespart. Da die Indexgröße kleiner ist, sind die Abfragen, die sie verwenden, effizienter. Damit eine Abfrage einen Index mit geringer Dichte verwendet, müssen Sie die $exists-Klausel für die indizierten Felder verwenden. Wenn Sie die $exists-Klausel weglassen, verwendet Amazon DocumentDB den Index mit geringer Dichte nicht.

Beispiele

Die folgenden Beispiele zeigen, wie Sie Indizes mit geringer Dichte für das folgende Beispieldokument erstellen:

```
{
  "productId": "PROD133726",
  "sku": "SKU24224",
  "name": "Basic Printer",
  "manufacturer": "The Manufacturer",
  "tags": [ "printer", "basic", "electronics", "business" ],
  "barcodes": [ "542364671", "886330670", "437445606" ],
  "reviews": [
    {
      "review_date": ISODate('2024-01-19T21:37:10.585Z'),
      ...
    }
  ],
  "material": "Polycarbonate",
  "color": "Space Gray",
  "supplier": {
    "supplierId": "SUP4",
    "location": {
      "type": "Point",
      "coordinates": [ -71.0589, 42.3601 ]
    }
  },
  "productEmbedding": [
```

```
    -0.019320633663838058,
    0.019672111388113596
  ],
  "lastUpdated": ISODate('2025-10-20T21:37:10.585Z')
}
```

Beachten Sie, dass die Felder „Bewertungen“, „Material“ und „Farbe“ nicht in allen Dokumenten vorhanden sind.

einzelnes Feld

Erstellen Sie einen dünn besetzten Einzelfeldindex für das Materialfeld:

```
db.collection.createIndex(
  {
    "material": 1
  },
  {
    "name": "material_sparse",
    "sparse": true
  }
)
```

Dieser Index wird verwendet, wenn alle Produkte gefunden werden, die einen Materialwert haben:

```
db.collection.find({
  "material": {
    $exists: true
  }
})
```

Verbindung

Erstellen Sie einen zusammengesetzten Index mit geringer Dichte für die Material- und Farbfelder:

```
db.collection.createIndex(
  {
    "material": 1,
    "color": 1
  },
  {
    "name": "material_and_color_sparse",
```

```
    "sparse": true
  }
)
```

Dieser Index wird verwendet, wenn alle Produkte gefunden werden, die eine beliebige Kombination von Material- und Farbwerten haben:

```
db.collection.find({
  "material": {
    $exists: true
  }
})

db.collection.find({
  "color": {
    $exists: true
  }
})

db.collection.find({
  $and: [
    {
      "material": {
        $exists: true
      }
    },
    {
      "color": {
        $exists:true
      }
    }
  ]
})

db.collection.find({
  $and: [
    {
      "color": {
        $exists: true
      }
    },
    {
      "material": {
```

```
        $exists:true
      }
    }
  ]
})
```

mehrere Schlüssel

Erstellen Sie einen Index mit mehreren Schlüsseln für das Reviews-Array mit geringer Dichte:

```
db.collection.createIndex(
  {
    "reviews": 1
  },
  {
    "name": "reviews_sparse",
    "sparse": true
  }
)
```

Dieser Index wird verwendet, wenn alle Produkte gefunden werden, die über ein Bewertungsfeld verfügen:

```
db.collection.find({
  "reviews": {
    $exists: true
  }
})
```

Index-Eigenschaft: teilweise FilterExpression

Unterstützte Indextypen

Option	3.6	4,0	5.0	8.0	Elastischer Cluster
einzelnes Feld	Nein	Nein	Ja	Ja	Nein
compound	Nein	Nein	Ja	Ja	Nein

Option	3.6	4,0	5.0	8.0	Elastischer Cluster
mehrere Schlüssel	Nein	Nein	Ja	Ja	Nein
Geodaten	Nein	Nein	Nein	Nein	Nein

Verwenden Sie die FilterExpression Option „Teilweise“, um einen Teilindex zu erstellen, der nur Dokumente enthält, die eine bestimmte Filterbedingung erfüllen. Auf diese Weise können Sie effizientere Indizes erstellen, indem Sie nur eine Teilmenge der Dokumente in einer Sammlung indexieren, anstatt alle Dokumente zu indexieren, wodurch die Indexgröße reduziert und Speicherplatz gespart wird. Da die Indexgröße kleiner ist, sind die Abfragen, die sie verwenden, effizienter. Amazon DocumentDB verwendet den Teilindex in den folgenden Szenarien:

- Das Abfrageprädikat entspricht exakt dem Filterausdruck für den Teilindex.
- Das erwartete Ergebnis des Abfragefilters ist eine logische Teilmenge des Teilfilters.
- Ein Unterprädikat der Abfrage kann in Verbindung mit anderen Indizes verwendet werden.

Weitere Informationen finden Sie unter Teilweiser Index.

Beispiele

Die folgenden Beispiele zeigen, wie Teilindizes für das folgende Beispieldokument erstellt werden:

```
{
  "productId": "PROD133726",
  "sku": "SKU24224",
  "name": "Basic Printer",
  "manufacturer": "The Manufacturer",
  "tags": [ "printer", "basic", "electronics", "business" ],
  "barcodes": [ "542364671", "886330670", "437445606" ],
  "reviews": [
    {
      "review_date": ISODate('2024-01-19T21:37:10.585Z'),
      ...
    }
  ],
  "material": "Polycarbonate",
```

```
  "color": "Space Gray",
  "supplier": {
    "supplierId": "SUP4",
    "location": {
      "type": "Point",
      "coordinates": [ -71.0589, 42.3601 ]
    }
  },
  "productEmbedding": [
    -0.019320633663838058,
    0.019672111388113596
  ],
  "lastUpdated": ISODate('2025-10-20T21:37:10.585Z')
}
```

einzelnes Feld

Erstellen Sie einen teilweisen Einzelfeldindex für den Hersteller für Produkte, die die Farbe Space Grey haben:

```
db.collection.createIndex(
  {
    "manufacturer": 1
  },
  {
    "name": "manufacturer_space_gray",
    "partialFilterExpression": {
      "color": {
        $eq: "Space Gray"
      }
    }
  }
)
```

Dieser Index wird bei der Suche nach Produkten in der Farbe Space Grey vom Hersteller verwendet:

```
db.collection.find({
  "manufacturer": "The Manufacturer",
  "color": {
    $eq: "Space Gray"
  }
})
```

Verbindung

Erstellen Sie einen partiellen Verbundindex für Hersteller und Farbe für Produkte, die aus Polycarbonat bestehen:

```
db.collection.createIndex(
  {
    "manufacturer": 1,
    "color": 1
  },
  {
    "name": "manufacturer_and_color_polycarbonate",
    "partialFilterExpression": {
      "material": {
        $eq: "Polycarbonate"
      }
    }
  }
)
```

Dieser Index wird bei der Suche nach Produkten aus Polycarbonat nach Hersteller und Farbe verwendet:

```
db.collection.find({
  "manufacturer": "The Manufacturer",
  "color": "Space Gray",
  "material": {
    $eq: "Polycarbonate"
  }
})
```

mit mehreren Tasten

Erstellen Sie einen teilweisen Index mit mehreren Schlüsseln für Tags für Produkte, für die der Hersteller als Hersteller angegeben ist:

```
db.collection.createIndex(
  {
    "tags": 1
  },
  {
```

```
    "name": "tags_the_manufacturer",
    "partialFilterExpression": {
      "manufacturer": {
        $eq: "The Manufacturer"
      }
    }
  }
)
```

Dieser Index wird bei der Suche nach Produkten verwendet, die mit einem Drucker gekennzeichnet sind und für die der Hersteller angegeben ist:

```
db.collection.find({
  "tags": "printer",
  "manufacturer": {
    $eq: "The Manufacturer"
  }
})
```

Indexeigenschaft: ablaufen AfterSeconds

Unterstützte Indextypen

Indextyp	3.6	4,0	5.0	8.0	Elastischer Cluster
einzelnes Feld	Ja	Ja	Ja	Ja	Ja
mehrere Schlüssel	Ja	Ja	Ja	Ja	Ja

Verwenden Sie die AfterSeconds Ablaufoption, um einen TTL-Index (Time to Live) zu erstellen. TTL-Indizes ermöglichen es Ihnen, Dokumente auf der Grundlage ihres Alters zu löschen, basierend auf der Timeout-Bedingung für jedes Dokument. Wenn ein Dokument das angegebene TTL-Alter erreicht, wird es aus der Sammlung gelöscht.

Bewährte Methoden für TTL-Löschungen finden Sie unter. TTL- und Zeitreihen-Workloads

Beispiele

Die folgenden Beispiele zeigen, wie TTL-Indizes für das folgende Beispieldokument erstellt werden:

```
{
  "productId": "PROD133726",
  "sku": "SKU24224",
  "name": "Basic Printer",
  "manufacturer": "The Manufacturer",
  "tags": [ "printer", "basic", "electronics", "business" ],
  "barcodes": [ "542364671", "886330670", "437445606" ],
  "reviews": [
    {
      "review_date": ISODate('2024-01-19T21:37:10.585Z'),
      "rating": 4,
      "comment": "Good product"
    },
    {
      "review_date": ISODate('2024-03-15T14:22:33.120Z'),
      "rating": 5,
      "comment": "Excellent printer"
    },
    {
      "review_date": ISODate('2024-06-08T09:45:18.890Z'),
      "rating": 3,
      "comment": "Average quality"
    }
  ],
  "material": "Polycarbonate",
  "color": "Space Gray",
  "supplier": {
    "supplierId": "SUP4",
    "location": {
      "type": "Point",
      "coordinates": [ -71.0589, 42.3601 ]
    }
  },
  "productEmbedding": [
    -0.019320633663838058,
    0.019672111388113596
  ],
  "lastUpdated": ISODate('2025-10-20T21:37:10.585Z')
}
```

einzelnes Feld

Erstellen Sie einen TTL-Index auf LastUpdated, um Dokumente zu löschen, die seit 90 Tagen nicht aktualisiert wurden:

```
db.collection.createIndex(
  {
    "lastUpdated": 1
  },
  {
    "name": "lastUpdated_ttl",
    "expireAfterSeconds": 7776000
  }
)
```

mehrere Schlüssel

Erstellen Sie einen TTL-Index auf reviews.review_date, um Dokumente zu löschen, für die im letzten Jahr keine Bewertungen eingegangen sind:

```
db.collection.createIndex(
  {
    "reviews.review_date": 1
  },
  {
    "name": "reviews_review_date_ttl",
    "expireAfterSeconds": 31536000
  }
)
```

Beachten Sie, dass, wenn sich der TTL-Index in einem Array-Feld befindet, alle Elemente im Array überprüft werden. Wenn ein Zeitstempel im Array die Timeout-Bedingung erfüllt, wird das Dokument gelöscht.

Indexeigenschaft: versteckt

Amazon DocumentDB unterstützt die `hidden` Indexeigenschaft ab Amazon DocumentDB 8.0.1+ und Amazon DocumentDB 5.0.2+.

Versteckte Indizes sind für den Abfrageplaner nicht sichtbar und können nicht zur Unterstützung einer Abfrage verwendet werden. Indem Sie einen Index im Planer ausblenden, können Sie die möglichen

Auswirkungen des Löschens des Indexes bewerten, ohne ihn tatsächlich zu löschen. Wenn die Auswirkung negativ ist, können Sie den Index wieder einblenden, anstatt einen gelöschten Index neu erstellen zu müssen.

Die `hidden` Eigenschaft wird für alle Indextypen unterstützt, einschließlich Einzelfeld-, Verbund-, Mehrschlüssel-, Text-, Geodaten- und Vektorindizes.

Behavior

Abgesehen davon, dass sie nicht im Planer angezeigt werden, werden versteckte Indizes vollständig verwaltet und durchgesetzt. Beispiel:

- Wenn es sich bei einem ausgeblendeten Index um einen eindeutigen Index handelt, wendet der Index dennoch seine eindeutige Einschränkung auf Dokumente an.
- Wenn es sich bei einem versteckten Index um einen TTL-Index handelt, läuft der Index dennoch für Dokumente ab.
- Versteckte Indizes sind in `listIndexes` und `db.collection.getIndexes()` Ergebnissen enthalten.
- Versteckte Indizes werden bei Schreibvorgängen in die Sammlung aktualisiert und verbrauchen weiterhin Speicherplatz und Arbeitsspeicher. Daher werden sie in Statistikoperationen wie `db.collection.stats()` und einbezogen`$indexStats`.
- Durch das Ausblenden eines nicht ausgeblendeten Indexes oder das Einblenden eines ausgeblendeten Indexes wird dessen Index zurückgesetzt. `$indexStats` Das Ausblenden eines bereits ausgeblendeten Indexes oder das Einblenden eines bereits ausgeblendeten Indexes wird nicht zurückgesetzt. `$indexStats`

Einschränkungen

- Der `_id` Index kann nicht ausgeblendet werden.
- Sie können ihn nicht verwenden`cursor.hint()`, um die Verwendung eines versteckten Indexes zu erzwingen.
- `$merge`schlägt fehl, wenn der erforderliche eindeutige Index für die on Felder ausgeblendet ist. Machen Sie den Index sichtbar, bevor Sie ihn ausführen`$merge`.

Beispiele

Erstellen Sie einen versteckten Index

Um einen Index im versteckten Zustand zu erstellen, verwenden Sie die `db.collection.createIndex()` Methode und setzen Sie die `hidden` Option `true` in der Indexdefinition auf.

Mit dem folgenden Vorgang wird beispielsweise ein versteckter aufsteigender Index für das `productId` Feld erstellt:

```
db.collection.createIndex(
  { "productId": 1 },
  { "name": "productId_1", "hidden": true }
)
```

Führen Sie zur Überprüfung `db.collection.getIndexes()` oder für `listIndexes` die Sammlung aus. Ein versteckter Index umfasst `"hidden": true` in seiner Definition Folgendes:

```
db.collection.getIndexes()
[
  {
    "v": 4,
    "key": { "_id": 1 },
    "name": "_id_"
  },
  {
    "v": 4,
    "key": { "productId": 1 },
    "name": "productId_1",
    "hidden": true
  }
]
```

Die Indexoption `hidden` wird nur zurückgegeben, wenn der Wert `true`

Einen vorhandenen Index ausblenden

Verwenden Sie den `collMod` Befehl, um einen vorhandenen Index auszublenden. Identifizieren Sie den Index entweder anhand seines `name` oder anhand seines`keyPattern`, und setzen Sie ihn `hidden` auf`true`.

Im folgenden Beispiel wird ein Index nach Namen ausgeblendet.

```
db.runCommand({
  collMod: "collection",
  index: {
    name: "productId_1", // Specify the index name
    hidden: true
  }
})
```

Sie können den Index auch anhand seines Schlüsselmusters identifizieren.

```
db.runCommand({
  collMod: "collection",
  index: {
    keyPattern: { productId: 1 }, // Specify the index key specification document
    hidden: true
  }
})
```

Blenden Sie einen vorhandenen Index ein

Um einen ausgeblendeten Index wieder sichtbar zu machen, verwenden Sie den `collMod` Befehl mit `hidden` set to`false`.

```
db.runCommand({
  collMod: "collection",
  index: { name: "productId_1", hidden: false }
})
```

Da Indizes vollständig verwaltet werden, wenn sie ausgeblendet sind, kann der Index sofort verwendet werden, sobald sie ausgeblendet sind.

Change Streams mit Amazon DocumentDB verwenden

Die Change-Streams-Funktion in Amazon DocumentDB (mit MongoDB-Kompatibilität) bietet eine zeitlich geordnete Abfolge von Änderungsereignissen, die in den Sammlungen Ihres Clusters auftreten. Sie können Ereignisse aus einem Change Stream lesen, um zahlreiche verschiedene Anwendungsfälle zu implementieren, einschließlich:

- Änderungsbenachrichtigung

- Full-text suche mit Amazon OpenSearch Service (Service) OpenSearch
- Analytik mit Amazon Redshift

Anwendungen können Änderungsstreams verwenden, um Datenveränderungen bei individuellen Sammlungen zu abonnieren. Die Ereignisse werden in Change Streams in der Reihenfolge angeordnet, wie sie im Cluster auftreten, und nach der Aufzeichnung des Ereignisses 3 Stunden (Standardeinstellung) gespeichert. Die Aufbewahrungsfrist kann mithilfe des `change_stream_log_retention_duration` Parameters auf bis zu 7 Tage verlängert werden. Informationen zum Ändern der Aufbewahrungsdauer von Change-Streams finden Sie unter Änderung der Aufbewahrungsdauer des Change-Stream-Protokolls.

Themen

- Unterstützte Vorgänge
- Erweiterte Ereignisse
- Fakturierung
- Einschränkungen
- Change Streams aktivieren
- Beispiel: Verwendung von Change Streams mit Python
- Vollständige Dokumentensuche
- Verwenden Sie $changeStreamSplitLargeEvent , um große Ereignisse in Fragmente aufzuteilen
- Wiederaufnahme eines Change-Streams
- Wiederaufnahme eines Change-Streams mit Start AtOperationTime
- Wiederaufnahme eines Change-Streams mit Post BatchResumeToken
- Transaktionen in Change-Streams
- Änderung der Aufbewahrungsdauer des Change-Stream-Protokolls
- Verwendung von Change Streams auf sekundären Instances

Unterstützte Vorgänge

Amazon DocumentDB unterstützt die folgenden Operationen für Change-Streams:

- Alle Änderungsereignisse werden in der MongoDB `db.watch()` und `client.watch()` der `db.collection.watch()` API unterstützt.

- Vollständige Dokumentsuche nach Aktualisierungen.
- Aggregationsstufen:`$match`,`$project`, `$redact` `$addFields``$replaceRoot`, und. `$changeStreamSplitLargeEvent`
- Wiederaufnahme eines Change-Streams von einem Resume-Token aus
- Wiederaufnahme eines Change-Streams anhand eines Zeitstempels mit `startAtOperation` (gilt für Amazon DocumentDB 4.0+)
- DDL-Ereignisbenachrichtigung für die `createCollection` Verwendung des `showExpandedEvents` Parameters (gilt für Amazon DocumentDB 8.0 ab Engine-Version 8.0.2).
- DDL-Ereignisbenachrichtigung für die `createIndexes` Verwendung des `showExpandedEvents` Parameters (gilt für Amazon DocumentDB 8.0 ab Engine-Version 8.0.2).

Erweiterte Ereignisse

> **Note**
>
> Erweiterte Ereignisse sind in Amazon DocumentDB 8.0 ab Engine-Version 8.0.2 verfügbar.

Change Streams unterstützen DDL-Ereignisbenachrichtigungen (Data Definition Language) und. `createCollection` `createIndexes` Um erweiterte Ereignisse einzubeziehen, öffnen Sie einen Change-Stream-Cursor mit der `showExpandedEvents` Option.

Beispiel:

mongosh Method

```
let cur = db.names.watch( [ ], {
   showExpandedEvents: true
} )
cur.next()
```

Aggregation Stage

```
let cur = db.names.aggregate( [
   { $changeStream: { showExpandedEvents: true } }
] )
cur.next()
```

Fakturierung

Die Amazon DocumentDB-Funktion für Change Streams ist standardmäßig deaktiviert und es fallen keine zusätzlichen Gebühren an, bis die Funktion aktiviert ist. Die Verwendung von Change Streams in einem Cluster verursacht zusätzliche I/Os- und Speicherkosten für Lese- und Schreibvorgänge. Sie können den `modifyChangeStreams` API-Vorgang verwenden, um diese Funktion für Ihren Cluster zu aktivieren. Weitere Informationen zur Preisgestaltung finden Sie unter Amazon DocumentDB-Preise.

Einschränkungen

Change Streams haben in Amazon DocumentDB die folgenden Einschränkungen:

- In Amazon DocumentDB 3.6. und Amazon DocumentDB 4.0 können Change-Streams nur über eine Verbindung zur primären Instance eines Amazon DocumentDB-Clusters geöffnet werden. Das Lesen von Change-Streams auf einer Replikat-Instance wird in Amazon DocumentDB 3.6. und Amazon DocumentDB 4.0 nicht unterstützt. Beim Aufruf der API-Operation `watch()` müssen Sie die Leseeinstellung `primary` angeben, um sicherzustellen, dass alle Lesevorgänge an die primäre Instance weitergeleitet werden (siehe den Abschnitt Beispiel).
- In Amazon DocumentDB 5.0 können Change-Streams sowohl von der primären als auch von der sekundären Instance aus geöffnet werden, einschließlich globaler Cluster. Sie können eine sekundäre Leseeinstellung angeben, um die Change-Streams an sekundäre Instances umzuleiten. Verwendung von Change Streams auf sekundären InstancesWeitere bewährte Methoden und Einschränkungen finden Sie unter.
- Ereignisse, die für eine Sammlung in einen Change-Stream geschrieben werden, sind bis zu 7 Tage lang verfügbar (die Standardeinstellung ist 3 Stunden). Änderungsstream-Daten werden nach dem Zeitfenster der Protokollaufbewahrungsdauer gelöscht, auch wenn keine neuen Änderungen vorgenommen wurden.
- Eine über längere Zeit für eine Sammlung ausgeführte Schreiboperation wie `updateMany` oder `deleteMany` kann das Schreiben von Change Stream-Ereignissen vorübergehend bis zum Abschluss der über längere Zeit ausgeführten Schreiboperation blockieren.
- Amazon DocumentDB unterstützt das MongoDB-Betriebsprotokoll () `oplog` nicht.
- Bei Amazon DocumentDB müssen Sie Change Streams für eine bestimmte Sammlung explizit aktivieren.
- Wenn die Gesamtgröße eines Change Streams-Ereignisses (einschließlich der Änderungsdaten und des vollständigen Dokuments, falls gewünscht) größer als ist`16 MB`, tritt beim Client

ein Lesefehler im Change-Stream auf. Um dies zu vermeiden, verwenden Sie den `$changeStreamSplitLargeEvent` Stagingbereich, um das Ereignis in Fragmente aufzuteilen.

- Der Ruby-Treiber wird derzeit nicht unterstützt, wenn `db.watch()` und `client.watch()` mit Amazon DocumentDB 3.6 verwendet wird.
- Die Ausgabe des `updateDescription` Befehls in Change Streams unterscheidet sich in Amazon DocumentDB von der in MongoDB, wenn der aktualisierte Wert des Felds dem vorherigen entspricht:
  - Amazon DocumentDB gibt kein Feld in der `updateDescription` Ausgabe zurück, wenn das angegebene Feld im `$set` Befehl angegeben ist und sein Zielwert bereits dem Quellwert entspricht.
  - MongoDB gibt das Feld in der Ausgabe zurück, auch wenn der angegebene Wert dem aktuellen Wert entspricht.

Change Streams aktivieren

Sie können Amazon DocumentDB-Change Streams für alle Sammlungen innerhalb einer bestimmten Datenbank oder nur für ausgewählte Sammlungen aktivieren. Im Folgenden finden Sie Beispiele für die Aktivierung von Change Streams für verschiedene Anwendungsfälle über die Mongo-Shell. Leere Zeichenfolgen werden bei der Angabe von Datenbank- und Sammlungsnamen als Platzhalter behandelt.

```
//Enable change streams for the collection "foo" in database "bar"
db.adminCommand({modifyChangeStreams: 1,
    database: "bar",
    collection: "foo",
    enable: true});
```

```
//Disable change streams on collection "foo" in database "bar"
db.adminCommand({modifyChangeStreams: 1,
    database: "bar",
    collection: "foo",
    enable: false});
```

```
//Enable change streams for all collections in database "bar"
db.adminCommand({modifyChangeStreams: 1,
    database: "bar",
    collection: "",
```

```
    enable: true});
```

```
//Enable change streams for all collections in all databases in a cluster
db.adminCommand({modifyChangeStreams: 1,
    database: "",
    collection: "",
    enable: true});
```

Change Streams werden für eine Sammlung aktiviert, wenn eine der folgenden Bedingungen erfüllt ist:

- Sowohl die Datenbank als auch die Sammlung sind explizit aktiviert.
- Die Datenbank, die die Sammlung enthält, ist aktiviert.
- Alle Datenbanken sind aktiviert.

Wenn Sie eine Sammlung aus einer Datenbank löschen, werden Change Streams für diese Sammlung nicht deaktiviert, wenn Change Streams für die übergeordnete Datenbank auch aktiviert sind oder wenn alle Datenbanken im Cluster aktiviert sind. Wenn eine neue Sammlung mit demselben Namen wie die gelöschte Sammlung erstellt wird, werden Change Streams für diese Sammlung aktiviert.

Sie können alle aktivierten Change Streams Ihres Clusters mithilfe der `$listChangeStreams`-Aggregationspipeline-Phase auflisten. Alle von Amazon DocumentDB unterstützten Aggregationsstufen können in der Pipeline für zusätzliche Verarbeitungsvorgänge verwendet werden. Wenn eine zuvor aktivierte Sammlung deaktiviert wurde, wird sie nicht in der `$listChangeStreams`-Ausgabe angezeigt.

```
//List all databases and collections with change streams enabled
cursor = new DBCommandCursor(db,
    db.runCommand(
        {aggregate: 1,
        pipeline: [{$listChangeStreams: 1}],
        cursor:{}}));
```

```
//List of all databases and collections with change streams enabled
{ "database" : "test", "collection" : "foo" }
{ "database" : "bar", "collection" : "" }
{ "database" : "", "collection" : "" }
```

```
//Determine if the database “bar” or collection “bar.foo” have change streams enabled
cursor = new DBCommandCursor(db,
  db.runCommand(
      {aggregate: 1,
       pipeline: [{$listChangeStreams: 1},
                  {$match: {$or: [{database: "bar", collection: "foo"},
                                  {database: "bar", collection: ""},
                                  {database: "", collection: ""}]}}
                ],
      cursor:{}}));
```

Beispiel: Verwendung von Change Streams mit Python

Im Folgenden finden Sie ein Beispiel für die Verwendung eines Amazon DocumentDB-Change-Streams mit Python auf Sammlungsebene.

```
import os
import sys
from pymongo import MongoClient, ReadPreference

username = "DocumentDBusername"
password = <Insert your password>

clusterendpoint = "DocumentDBClusterEndpoint”
client = MongoClient(clusterendpoint, username=username, password=password, tls='true',
 tlsCAFile='global-bundle.pem')

db = client['bar']

#While ‘Primary’ is the default read preference, here we give an example of
#how to specify the required read preference when reading the change streams
coll = db.get_collection('foo', read_preference=ReadPreference.PRIMARY)
#Create a stream object
stream = coll.watch()
#Write a new document to the collection to generate a change event
coll.insert_one({'x': 1})
#Read the next change event from the stream (if any)
print(stream.try_next())

"""
Expected Output:
{'_id': {'_data': '015daf94f600000002010000000200009025'},
```

```
'clusterTime': Timestamp(1571788022, 2),
'documentKey': {'_id': ObjectId('5daf94f6ea258751778163d6')},
'fullDocument': {'_id': ObjectId('5daf94f6ea258751778163d6'), 'x': 1},
'ns': {'coll': 'foo', 'db': 'bar'},
'operationType': 'insert'}
"""

#A subsequent attempt to read the next change event returns nothing, as there are no
 new changes
print(stream.try_next())

"""
Expected Output:
None
"""

#Generate a new change event by updating a document
result = coll.update_one({'x': 1}, {'$set': {'x': 2}})
print(stream.try_next())

"""
Expected Output:
{'_id': {'_data': '015daf99d400000001010000000100009025'},
'clusterTime': Timestamp(1571789268, 1),
'documentKey': {'_id': ObjectId('5daf9502ea258751778163d7')},
'ns': {'coll': 'foo', 'db': 'bar'},
'operationType': 'update',
'updateDescription': {'removedFields': [], 'updatedFields': {'x': 2}}}
"""
```

Das Folgende ist ein Beispiel für die Verwendung eines Amazon DocumentDB-Change-Streams mit Python auf Datenbankebene.

```
import os
import sys
from pymongo import MongoClient

username = "DocumentDBusername"
password = <Insert your password>
clusterendpoint = "DocumentDBClusterEndpoint”
client = MongoClient(clusterendpoint, username=username, password=password, tls='true',
 tlsCAFile='global-bundle.pem')
```

```
db = client['bar']
#Create a stream object
stream = db.watch()
coll = db.get_collection('foo')
#Write a new document to the collection foo to generate a change event
coll.insert_one({'x': 1})

#Read the next change event from the stream (if any)
print(stream.try_next())

"""
Expected Output:
{'_id': {'_data': '015daf94f600000002010000000200009025'},
'clusterTime': Timestamp(1571788022, 2),
'documentKey': {'_id': ObjectId('5daf94f6ea258751778163d6')},
'fullDocument': {'_id': ObjectId('5daf94f6ea258751778163d6'), 'x': 1},
'ns': {'coll': 'foo', 'db': 'bar'},
'operationType': 'insert'}
"""
#A subsequent attempt to read the next change event returns nothing, as there are no
 new changes
print(stream.try_next())

"""
Expected Output:
None
"""

coll = db.get_collection('foo1')

#Write a new document to another collection to generate a change event
coll.insert_one({'x': 1})
print(stream.try_next())

"""
Expected Output: Since the change stream cursor was the database level you can see
 change events from different collections in the same database
{'_id': {'_data': '015daf94f600000002010000000200009025'},
'clusterTime': Timestamp(1571788022, 2),
'documentKey': {'_id': ObjectId('5daf94f6ea258751778163d6')},
'fullDocument': {'_id': ObjectId('5daf94f6ea258751778163d6'), 'x': 1},
'ns': {'coll': 'foo1', 'db': 'bar'},
'operationType': 'insert'}
```

```
"""
```

Vollständige Dokumentensuche

Das Änderungsereignis „Aktualisierung“ enthält nicht das vollständige Dokument, sondern lediglich die ausgeführte Änderung. Wenn für Ihren Anwendungsfall das vollständige Dokument erforderlich ist, das von einer Aktualisierung betroffen ist, können Sie beim Öffnen des Datenstroms die vollständige Dokumentsuche aktivieren.

Das `fullDocument`-Dokument für ein Update-Change Stream-Ereignis stellt die neueste Version des aktualisierten Dokuments zum Zeitpunkt der Dokumentsuche dar. Wenn zwischen der Aktualisierungsoperation und der `fullDocument`-Suche Änderungen ausgeführt wurden, besitzt das `fullDocument`-Dokument möglicherweise nicht den Dokumentstatus zur Aktualisierungszeit.

Verwenden Sie dieses Beispiel, um ein Stream-Objekt mit aktivierter Aktualisierungssuche zu erstellen:

```
stream = coll.watch(full_document='updateLookup')

#Generate a new change event by updating a document
result = coll.update_one({'x': 2}, {'$set': {'x': 3}})

stream.try_next()
```

Die Ausgabe des Stream-Objekts sieht ungefähr so aus:

```
{'_id': {'_data': '015daf9b7c000000010100000100009025'},
'clusterTime': Timestamp(1571789692, 1),
'documentKey': {'_id': ObjectId('5daf9502ea258751778163d7')},
'fullDocument': {'_id': ObjectId('5daf9502ea258751778163d7'), 'x': 3},
'ns': {'coll': 'foo', 'db': 'bar'},
'operationType': 'update',
'updateDescription': {'removedFields': [], 'updatedFields': {'x': 3}}}
```

Verwenden Sie $changeStreamSplitLargeEvent , um große Ereignisse in Fragmente aufzuteilen

> **Note**
>
> Die StreamSplitLargeEvent $change-Stufe ist in Amazon DocumentDB 8.0 ab Engine-Version 8.0.2 verfügbar.

Wenn Sie den `$changeStreamSplitLargeEvent` Stagingbereich verwenden, werden Änderungsereignisse, die größer als 16 MB sind, in Fragmente aufgeteilt, die nacheinander über den Change-Stream-Cursor zurückgegeben werden. Ereignisse, die kleiner als 16 MB sind, werden ungeteilt zurückgegeben. Die Phase teilt ein Ereignis entlang seiner Feldgrenzen auf oberster Ebene auf. Es wird nicht innerhalb eines einzelnen Felds aufgeteilt und auch nicht in Unterdokumente aufgeteilt. Aus diesem Grund kann der Schritt, wenn ein einzelnes Feld der obersten Ebene selbst größer als 16 MB ist, kein gültiges Fragment dafür erstellen und gibt stattdessen einen Fehler zurück.

Als bewährte Methode empfehlen wir, die Verwendung von möglichst zu vermeiden`$changeStreamSplitLargeEvent`, indem Sie die Größe des Ereignisses reduzieren.

Die `$changeStreamSplitLargeEvent` Stufe kann nur in einer `$changeStream` Pipeline verwendet werden, und sie darf nur einmal als letzte Stufe verwendet werden.

Jedes Fragment hat sein eigenes `resumeToken` und sein eigenes`postBatchResumeToken`. Wenn Sie von einem Fragment-Token aus weitermachen, das noch nicht das letzte Fragment ist (entweder mit einem regulären Resume-Token oder mit einem PostBatchResumeToken), wird das nachfolgende Fragment generiert. Falls das Token dem letzten Fragment eines Ereignisses entspricht, gibt der Change-Stream-Cursor das nachfolgende Ereignis zurück.

Wiederaufnahme eines Change-Streams

Sie können einen Change Stream zu einem späteren Zeitpunkt mithilfe eines Fortsetzungs-Tokens fortsetzen, das dem Feld `_id` des zuletzt abgerufenen Änderungsereignisdokuments entspricht.

```
import os
import sys
from pymongo import MongoClient

username = "DocumentDBusername"
```

```
password = <Insert your password>
clusterendpoint = "DocumentDBClusterEndpoint”
client = MongoClient(clusterendpoint, username=username, password=password, tls='true',
 tlsCAFile='global-bundle.pem', retryWrites='false')

db = client['bar']
coll = db.get_collection('foo')
#Create a stream object
stream = db.watch()
coll.update_one({'x': 1}, {'$set': {'x': 4}})
event = stream.try_next()
token = event['_id']
print(token)

"""
Output: This is the resume token that we will later us to resume the change stream
{'_data': '015daf9c5b0000000101000000100009025'}
"""
#Python provides a nice shortcut for getting a stream’s resume token
print(stream.resume_token)

"""
Output
{'_data': '015daf9c5b0000000101000000100009025'}
"""
#Generate a new change event by updating a document
result = coll.update_one({'x': 4}, {'$set': {'x': 5}})
#Generate another change event by inserting a document
result = coll.insert_one({'y': 5})
#Open a stream starting after the selected resume token
stream = db.watch(full_document='updateLookup', resume_after=token)
#Our first change event is the update with the specified _id
print(stream.try_next())

"""
#Output: Since we are resuming the change stream from the resume token, we will see all
 events after the first update operation. In our case, the change stream will resume
 from the update operation {x:5}

{'_id': {'_data': '015f7e8f0c0000000601000000006000fe038'},
'operationType': 'update',
'clusterTime': Timestamp(1602129676, 6),
'ns': {'db': 'bar', 'coll': 'foo'},
'documentKey': {'_id': ObjectId('5f7e8f0ac423bafbfd9adba2')},
```

```
'fullDocument': {'_id': ObjectId('5f7e8f0ac423bafbfd9adba2'), 'x': 5},
'updateDescription': {'updatedFields': {'x': 5}, 'removedFields': []}}
"""
#Followed by the insert
print(stream.try_next())

"""
#Output:
{'_id': {'_data': '015f7e8f0c0000000701000000007000fe038'},
'operationType': 'insert',
'clusterTime': Timestamp(1602129676, 7),
'ns': {'db': 'bar', 'coll': 'foo'},
'documentKey': {'_id': ObjectId('5f7e8f0cbf8c233ed577eb94')},
'fullDocument': {'_id': ObjectId('5f7e8f0cbf8c233ed577eb94'), 'y': 5}}
"""
```

Wiederaufnahme eines Change-Streams mit Start **`AtOperationTime`**

Sie können einen Change-Stream später ab einem bestimmten Zeitstempel fortsetzen, indem Sie Folgendes verwenden`startAtOperationTime`.

> **Note**
>
> Die Möglichkeit zur Verwendung `startAtOperationTime` ist in Amazon DocumentDB 4.0+ verfügbar. Bei Verwendung `startAtOperationTime` gibt der Change-Stream-Cursor nur Änderungen zurück, die zum oder nach dem angegebenen Zeitstempel vorgenommen wurden. Die `resumeAfter` Befehle `startAtOperationTime` und schließen sich gegenseitig aus und können daher nicht zusammen verwendet werden.

```
import os
import sys
from pymongo import MongoClient

username = "DocumentDBusername"
password = <Insert your password>
clusterendpoint = "DocumentDBClusterEndpoint”
client = MongoClient(clusterendpoint, username=username, password=password, tls='true',
 tlsCAFile='rds-root-ca-2020.pem',retryWrites='false')
db = client['bar']
coll = db.get_collection('foo')
```

```
#Create a stream object
stream = db.watch()
coll.update_one({'x': 1}, {'$set': {'x': 4}})
event = stream.try_next()
timestamp = event['clusterTime']
print(timestamp)
"""
Output
Timestamp(1602129114, 4)
"""
#Generate a new change event by updating a document
result = coll.update_one({'x': 4}, {'$set': {'x': 5}})
result = coll.insert_one({'y': 5})
#Generate another change event by inserting a document
#Open a stream starting after specified time stamp

stream = db.watch(start_at_operation_time=timestamp)
print(stream.try_next())

"""
#Output: Since we are resuming the change stream at the time stamp of our first update
 operation (x:4), the change stream cursor will point to that event
{'_id': {'_data': '015f7e941a0000000301000000003000fe038'},
'operationType': 'update',
'clusterTime': Timestamp(1602130970, 3),
'ns': {'db': 'bar', 'coll': 'foo'},
'documentKey': {'_id': ObjectId('5f7e9417c423bafbfd9adbb1')},
'updateDescription': {'updatedFields': {'x': 4}, 'removedFields': []}}
"""

print(stream.try_next())
"""
#Output: The second event will be the subsequent update operation (x:5)
{'_id': {'_data': '015f7e95020000000501000000005000fe038'},
'operationType': 'update',
'clusterTime': Timestamp(1602131202, 5),
'ns': {'db': 'bar', 'coll': 'foo'},
'documentKey': {'_id': ObjectId('5f7e94ffc423bafbfd9adbb2')},
'updateDescription': {'updatedFields': {'x': 5}, 'removedFields': []}}
"""

print(stream.try_next())

"""
```

```
#Output: And finally the last event will be the insert operation (y:5)
{'_id': {'_data': '015f7e9502000000060100000006000fe038'},
'operationType': 'insert',
'clusterTime': Timestamp(1602131202, 6),
'ns': {'db': 'bar', 'coll': 'foo'},
'documentKey': {'_id': ObjectId('5f7e95025c4a569e0f6dde92')},
'fullDocument': {'_id': ObjectId('5f7e95025c4a569e0f6dde92'), 'y': 5}}
"""
```

Wiederaufnahme eines Change-Streams mit Post **`BatchResumeToken`**

Amazon DocumentDB Change Stream gibt jetzt ein zusätzliches Feld mit dem Namen zurück. `postBatchResumeToken` Dieses Feld wird aus dem `$changestream` Befehl und `getMore` dem Befehl zurückgegeben.

Beispiel für den `$changestream` Befehl in Python:

```
db.command({"aggregate": "sales", "pipeline": [{ "$changeStream": {}}], "cursor":
 {"batchSize": 1}})
```

Erwartete Ausgabe:

```
cursor" : {
   "firstBatch" : [ ],
   "postBatchResumeToken" : {"_data" : "0167c8cbe60000000004"},
   "id" : NumberLong("9660788144470"),
   "ns" : "test.sales"
}
```

Beispiel für den `getMore` Befehl in Python:

```
db.command({"getMore": NumberLong(<cursor id>), "collection": "sales", "batchSize":
 1 })
```

Erwartete Ausgabe

```
cursor" : {
   "nextBatch" : [ ],
   "postBatchResumeToken" : {"_data" : "0167c8cbe60000000004"},
```

```
    "id" : NumberLong("9660788144470"),
    "ns" : "test.sales"
}
```

Das `postBatchResumeToken` Feld kann verwendet werden, um neue Change-Stream-Cursor im `resumeAfter` Feld zu öffnen, ähnlich wie das Resume-Token verwendet wird.

Öffnet einen Stream, der nach dem ausgewählten `postBatchResumeToken` beginnt:

```
post_batch_resume_token = output['cursor']['postBatchResumeToken']
stream = db.watch(full_document='updateLookup', resume_after=post_batch_resume_token)
```

Im Gegensatz zu einem regulären Resume-Token, das immer einem Operations-Log-Eintrag (Oplog) entspricht, der ein aktuelles Ereignis widerspiegelt, `postBatchResumeToken` entspricht es einem Oplog-Eintrag, bis zu dem der Change-Stream auf dem Server gescannt hat, was nicht unbedingt eine entsprechende Änderung ist.

Der Versuch, mit einem alten regulären Wiederaufnahme-Token fortzufahren, zwingt die Datenbank, alle Oplog-Einträge zwischen dem angegebenen Zeitstempel und der aktuellen Uhrzeit zu scannen. Dadurch können intern viele Abfragen generiert werden, wobei jede Unterabfrage für einen kurzen Zeitraum gescannt wird. Dies führt zu einem Anstieg der CPU-Auslastung und beeinträchtigt die Datenbankleistung. Wenn Sie mit dem letzten fortfahren, wird das Scannen nicht übereinstimmender `postBatchResumeToken` Oplog-Einträge übersprungen.

Transaktionen in Change-Streams

Change-Stream-Ereignisse enthalten keine Ereignisse aus nicht bestätigten and/or abgebrochenen Transaktionen. Wenn Sie beispielsweise eine Transaktion mit einer Operation und einer `INSERT` Operation starten und Ihr `UPDATE` Vorgang erfolgreich ist, der `INSERT` `UPDATE` Vorgang jedoch fehlschlägt, wird die Transaktion zurückgesetzt. Da für diese Transaktion ein Rollback durchgeführt wurde, enthält Ihr Change-Stream keine Ereignisse für diese Transaktion.

Änderung der Aufbewahrungsdauer des Change-Stream-Protokolls

Sie können die Aufbewahrungsdauer des Change-Stream-Protokolls so ändern, dass sie zwischen 1 Stunde und 7 Tagen liegt, indem Sie die AWS-Managementkonsole oder verwenden AWS CLI.

Using the AWS-Managementkonsole

So ändern Sie den Aufbewahrungszeitraum für das Change-Stream-Protokoll

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Wählen Sie im Navigationsbereich Parameter groups (Parametergruppen) aus.

   > **Tip**
   >
   > Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Wählen Sie im Bereich Parameter groups (Parametergruppen) die Cluster-Parametergruppe, die Ihrem Cluster zugeordnet ist. Informationen zum Identifizieren der Clusterparametergruppe, die dem Cluster zugeordnet ist, finden Sie unter Bestimmung der Parametergruppe eines Amazon DocumentDB-Clusters.
4. Auf der resultierenden Seite werden die Parameter und die entsprechenden Details zu Ihrer Cluster-Parametergruppe angezeigt. Wählen Sie den `change_stream_log_retention_duration`-Parameter aus.
5. Klicken Sie oben rechts auf der Seite auf Edit (Bearbeiten), um den Wert des Parameters zu ändern. Der `change_stream_log_retention_duration` Parameter kann so geändert werden, dass er zwischen 1 Stunde und 7 Tagen liegt.
6. Führen Sie die Änderung aus. Wählen Sie anschließend Modify cluster parameter (Cluster-Parameter ändern) aus, um die Änderungen zu speichern. Um die Änderungen zu verwerfen, wählen Sie Cancel (Abbrechen) aus.

Using the AWS CLI

Um den `change_stream_log_retention_duration`-Parameter Ihrer Cluster-Parametergruppe zu ändern, verwenden Sie die `modify-db-cluster-parameter-group`-Operation mit den folgenden Parametern:

- **`--db-cluster-parameter-group-name`** – Erforderlich. Der Name der Cluster-Parametergruppe, die Sie ändern. Informationen zum Identifizieren der

Clusterparametergruppe, die dem Cluster zugeordnet ist, finden Sie unter Bestimmung der Parametergruppe eines Amazon DocumentDB-Clusters.

- **`--parameters`** – Erforderlich. Der Parameter, der von Ihnen geändert wird. Jeder Parametereintrag muss Folgendes enthalten:
  - **`ParameterName`**— Der Name des Parameters, den Sie ändern. In diesem Fall ist es `change_stream_log_retention_duration`
  - **`ParameterValue`**— Der neue Wert für diesen Parameter.
  - **`ApplyMethod`**— Wie Sie Änderungen an diesem Parameter anwenden möchten. Zugelassene Werte sind `immediate` und `pending-reboot`.

> **Note**
>
> Parameter mit dem `ApplyType` von `static` müssen über einen `ApplyMethod` von `pending-reboot` verfügen.

1. Um die Werte des Parameters `change_stream_log_retention_duration` zu ändern, führen Sie den folgenden Befehl aus und ersetzen `parameter-value` durch den Wert, auf den Sie den Parameter ändern möchten.

   Für Linux, macOS oder Unix:

   ```
   aws docdb modify-db-cluster-parameter-group \
       --db-cluster-parameter-group-name sample-parameter-group \
       --parameters
    "ParameterName=change_stream_log_retention_duration,ParameterValue=<parameter-
   value>,ApplyMethod=immediate"
   ```

   Für Windows:

   ```
   aws docdb modify-db-cluster-parameter-group ^
       --db-cluster-parameter-group-name sample-parameter-group ^
       --parameters
    "ParameterName=change_stream_log_retention_duration,ParameterValue=<parameter-
   value>,ApplyMethod=immediate"
   ```

   Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    "DBClusterParameterGroupName": "sample-parameter-group"
}
```

2. Warten Sie mindestens 5 Minuten.
3. Listen Sie die Parameterwerte von `sample-parameter-group` auf, um sicherzustellen, dass Ihre Änderungen übernommen wurden.

   Für Linux, macOS oder Unix:

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

   Für Windows:

```
aws docdb describe-db-cluster-parameters ^
    --db-cluster-parameter-group-name sample-parameter-group
```

   Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    "Parameters": [
        {
            "ParameterName": "audit_logs",
            "ParameterValue": "disabled",
            "Description": "Enables auditing on cluster.",
            "Source": "system",
            "ApplyType": "dynamic",
            "DataType": "string",
            "AllowedValues": "enabled,disabled",
            "IsModifiable": true,
            "ApplyMethod": "pending-reboot"
        },
        {
            "ParameterName": "change_stream_log_retention_duration",
            "ParameterValue": "12345",
            "Description": "Duration of time in seconds that the change stream
 log is retained and can be consumed.",
            "Source": "user",
            "ApplyType": "dynamic",
```

```
            "DataType": "integer",
            "AllowedValues": "3600-86400",
            "IsModifiable": true,
            "ApplyMethod": "immediate"
        }
    ]
}
```

> **Note**
>
> Bei der Aufbewahrung von Change-Stream-Protokollen werden Protokolle, die älter als der konfigurierte `change_stream_log_retention_duration` Wert sind, erst gelöscht, wenn die Protokollgröße größer als (>) 51.200 MB ist.

Verwendung von Change Streams auf sekundären Instances

Um mit der Verwendung von Change Stream auf sekundären Instanzen zu beginnen, öffnen Sie den Change-Stream-Cursor mit dem `readPreference` Wert als sekundäre Instanz.

Sie können einen Change-Stream-Cursor öffnen, um auf Änderungsereignisse für eine bestimmte Sammlung oder alle Sammlungen in einem Cluster oder einer Datenbank zu achten. Sie können einen Change-Stream-Cursor auf jeder Amazon DocumentDB-Instance öffnen und Change-Stream-Dokumente sowohl von Writer- als auch von Reader-Instances abrufen. Sie können Change-Stream-Token (wie `resumeToken` oder`startOperationTime`) für verschiedene Change-Stream-Cursor gemeinsam nutzen, die auf einer Writer- und Reader-Instance geöffnet sind.

Beispiel

```
import os
import sys
from pymongo import MongoClient, ReadPreference

username = "DocumentDBusername"
password = <Your password>

clusterendpoint = "DocumentDBClusterEndpoint"

client = MongoClient(clusterendpoint, username=username, password=password, tls='true',
 tlsCAFile='global-bundle.pem')
```

```
db = client['bar']

# Make sure to use SECONDARY to redirect cursor reads from secondary instances
coll = db.get_collection('foo', read_preference=ReadPreference.SECONDARY)

# Create a stream object on RO. The token needs to generated from PRIMARY.
stream = coll.watch(resumeAfter=token)

for event in stream:
   print(event)
```

Richtlinien und Einschränkungen für Change-Streams auf sekundären Instances

- Change-Stream-Ereignisse müssen von der primären Instance auf die sekundären Instances repliziert werden. Sie können die Verzögerung anhand der `DBInstanceReplicaLag` Metrik in Amazon überwachen CloudWatch.
- Zeitstempel auf sekundären Instances stimmen möglicherweise nicht immer mit der primären Instance überein. In diesem Fall müssen Sie mit Verzögerungen beim Zeitstempel der sekundären Instance rechnen, damit diese aufholen kann. Es hat sich bewährt, `startAtOperationTime` oder `resumeToken` zu verwenden, um die Uhr auf der sekundären Instanz zu starten.
- Auf sekundären Instanzen kann es im Vergleich zur primären Instanz zu einem geringeren Durchsatz kommen, wenn Sie gerade `fullDocumentLookup` ein großes Dokument haben und auf der primären Instanz eine hohe Arbeitslast beim gleichzeitigen Schreiben anfällt. Es hat sich bewährt, die Trefferquote im Puffercache auf der Sekundärseite zu überwachen und sicherzustellen, dass die Trefferquote im Puffercache hoch ist.

Transaktionen in Amazon DocumentDB

Amazon DocumentDB (mit MongoDB-Kompatibilität) unterstützt Transaktionen in Amazon DocumentDB 4.0 und höher. Sie können Transaktionen über mehrere Dokumente, Kontoauszüge, Sammlungen und Datenbanken hinweg durchführen. Transaktionen vereinfachen die Anwendungsentwicklung, indem sie es Ihnen ermöglichen, atomare, konsistente, isolierte und dauerhafte (ACID) -Operationen für ein oder mehrere Dokumente innerhalb eines Amazon DocumentDB-Clusters durchzuführen. Zu den häufigsten Anwendungsfällen für Transaktionen gehören die Finanzabwicklung, die Erfüllung und Verwaltung von Bestellungen und die Entwicklung von Spielen für mehrere Spieler.

Für Transaktionen fallen keine zusätzlichen Kosten an. Sie zahlen nur für die Lese- und Schreib-IOs, die Sie im Rahmen der Transaktionen verbrauchen.

Themen

- Voraussetzungen
- Bewährte Methoden
- Einschränkungen
- Überwachung und Diagnose
- Isolationsstufe der Transaktion
- Anwendungsfälle
- Unterstützte Befehle
- Nicht unterstützte Funktionen
- Sitzungen
- Fehler bei der Transaktion

Voraussetzungen

Um die Transaktionsfunktion nutzen zu können, müssen Sie die folgenden Anforderungen erfüllen:

- Sie müssen Amazon DocumentDB 4.0 oder höher verwenden.
- Sie müssen einen Treiber verwenden, der mit MongoDB 4.0 oder höher kompatibel ist.

Bewährte Methoden

Im Folgenden finden Sie einige bewährte Methoden, damit Sie Transaktionen mit Amazon DocumentDB optimal nutzen können.

- Bestätigen Sie die Transaktion immer oder brechen Sie sie ab, nachdem sie abgeschlossen ist. Wenn Sie eine Transaktion in einem unvollständigen Zustand belassen, werden Datenbankressourcen gebunden und es kann zu Schreibkonflikten kommen.
- Es wird empfohlen, Transaktionen auf die kleinste Anzahl von benötigten Befehlen zu beschränken. Wenn Sie Transaktionen mit mehreren Kontoauszügen haben, die in mehrere kleinere Transaktionen aufgeteilt werden können, ist es ratsam, dies zu tun, um die Wahrscheinlichkeit eines Timeouts zu verringern. Versuchen Sie immer, kurze Transaktionen zu erstellen, keine lang andauernden Lesevorgänge.

Einschränkungen

- Amazon DocumentDB unterstützt keine Cursor innerhalb einer Transaktion.
- Amazon DocumentDB kann in einer Transaktion keine neuen Sammlungen erstellen und auch nicht query/update gegen nicht vorhandene Sammlungen abgleichen.
- Document-level Für Schreibsperren gilt ein Timeout von 1 Minute, das vom Benutzer nicht konfiguriert werden kann.
- In Engine-Versionen vor 8.0.2 werden wiederholbare Schreibvorgänge nicht unterstützt. Ab Engine-Version 8.0.2 werden wiederholbare Schreibvorgänge außerhalb von Transaktionen für Schreibvorgänge unterstützt, bei denen sich jeder einzelne Schreibvorgang auf höchstens ein Dokument auswirkt und Transaktions-Commit- und Abbruchoperationen wiederholt werden können. Weitere Informationen finden Sie unter Wiederholbare Schreibvorgänge in Amazon DocumentDB.
- Jede Amazon DocumentDB-Instance hat eine Obergrenze für die Anzahl gleichzeitiger Transaktionen, die auf der Instance geöffnet werden können. Weitere Informationen finden Sie unter Kontingente für Instanzen.
- Für eine bestimmte Transaktion muss die Größe des Transaktionsprotokolls weniger als 32 MB betragen.
- Amazon DocumentDB unterstützt zwar Transaktionen `count()` innerhalb einer Transaktion, aber nicht alle Treiber unterstützen diese Funktion. Eine Alternative ist die Verwendung der `countDocuments()` API, die die Zählabfrage in eine Aggregationsabfrage auf der Clientseite übersetzt.
- Transaktionen haben ein Ausführungslimit von einer Minute und Sitzungen haben ein Timeout von 30 Minuten. Wenn bei einer Transaktion ein Timeout auftritt, wird sie abgebrochen, und alle nachfolgenden Befehle, die innerhalb der Sitzung für die bestehende Transaktion ausgegeben werden, führen zu dem folgenden Fehler:

```
WriteCommandError({
"ok" : 0,
"operationTime" : Timestamp(1603491424, 627726),
"code" : 251,
"errmsg" : "Given transaction number 0 does not match any in-progress transactions."
}
```

Überwachung und Diagnose

Mit der Unterstützung für Transaktionen in Amazon DocumentDB 4.0 wurden zusätzliche CloudWatch Metriken hinzugefügt, die Sie bei der Überwachung Ihrer Transaktionen unterstützen.

Neue Metriken CloudWatch

- `DatabaseTransactions`: Die Anzahl der offenen Transaktionen, die in einem Zeitraum von einer Minute getätigt wurden.
- `DatabaseTransactionsAborted`: Die Anzahl der abgebrochenen Transaktionen, die innerhalb eines Zeitraums von einer Minute getätigt wurden.
- `DatabaseTransactionsMax`: Die maximale Anzahl offener Transaktionen in einem Zeitraum von einer Minute.
- `TransactionsAborted`: Die Anzahl der Transaktionen, die auf einer Instance innerhalb eines Zeitraums von einer Minute abgebrochen wurden.
- `TransactionsCommitted`: Die Anzahl der Transaktionen, die auf einer Instance innerhalb eines Zeitraums von einer Minute ausgeführt wurden.
- `TransactionsOpen`: Die Anzahl der auf einer Instance geöffneten Transaktionen, die innerhalb eines Zeitraums von einer Minute ausgeführt wurden.
- `TransactionsOpenMax`: Die maximale Anzahl von Transaktionen, die auf einer Instance innerhalb eines Zeitraums von einer Minute geöffnet werden.
- `TransactionsStarted`: Die Anzahl der Transaktionen, die auf einer Instance innerhalb eines Zeitraums von einer Minute gestartet wurden.

> **Note**
>
> Weitere CloudWatch Metriken für Amazon DocumentDB finden Sie unter. Überwachung von Amazon DocumentDB mit CloudWatch

Darüber hinaus wurden beiden Feldern `currentOplsid`,`transactionThreadId`, und ein neuer Status für „`idle transaction`“ und `serverStatus` Transaktionen hinzugefügt: `currentActive` `currentInactivecurrentOpen`, `totalAbortedtotalCommitted`, und`totalStarted`.

Isolationsstufe der Transaktion

Wenn Sie eine Transaktion starten, haben Sie die Möglichkeit, `readConcern` sowohl die als auch anzugeben, `writeConcern` wie im folgenden Beispiel gezeigt:

```
mySession.startTransaction({readConcern: {level: 'snapshot'}, writeConcern:
{w: 'majority'}});
```

Denn `readConcern` Amazon DocumentDB unterstützt standardmäßig die Snapshot-Isolierung. Wenn „Lokal“, „Verfügbar“ oder „Mehrheit“ angegeben ist, aktualisiert Amazon DocumentDB das `readConcern` Level auf Snapshot. `readConcern` Amazon DocumentDB unterstützt das Linearisierbare nicht `readConcern` und die Angabe eines solchen Leseproblems führt zu einem Fehler.

Amazon DocumentDB unterstützt standardmäßig die Mehrheit`writeConcern`, und ein Schreibquorum wird erreicht, wenn vier Kopien der Daten in drei AZs gespeichert sind. Wenn ein niedrigerer Wert angegeben `writeConcern` ist, aktualisiert Amazon DocumentDB den Wert auf den Mehrheitswert. `writeConcern` Außerdem werden alle Amazon DocumentDB-Schreibvorgänge protokolliert, und das Journaling kann nicht deaktiviert werden.

Anwendungsfälle

In diesem Abschnitt gehen wir auf zwei Anwendungsfälle für Transaktionen ein: Transaktionen mit mehreren Kontoauszügen und Mehrfacherfassung.

Multi-Statement Transaktionen

Amazon DocumentDB-Transaktionen bestehen aus mehreren Anweisungen, was bedeutet, dass Sie eine Transaktion schreiben können, die sich über mehrere Anweisungen erstreckt, mit einem expliziten Commit oder Rollback. Sie können `findAndModify` Aktionen`insert`, `update``delete`, und als eine einzelne atomare Operation gruppieren.

Ein häufiger Anwendungsfall für Transaktionen mit mehreren Kontoauszügen ist eine Last-/Kredittransaktion. Zum Beispiel: Du schuldest einem Freund Geld für Kleidung. Daher musst du 500$ von deinem Konto abbuchen (abheben) und dem Konto deines Freundes 500$ (Einzahlung) gutschreiben. Um diesen Vorgang durchzuführen, führst du sowohl den Lastschriftvorgang als auch den Kreditvorgang innerhalb einer einzigen Transaktion aus, um die Atomarität sicherzustellen. Dadurch werden Szenarien vermieden, in denen 500$ von deinem Konto abgebucht, aber nicht dem Konto deines Freundes gutgeschrieben werden. So würde dieser Anwendungsfall aussehen:

```
// *** Transfer $500 from Alice to Bob inside a transaction: Success Scenario***
// Setup bank account for Alice and Bob. Each have $1000 in their account

var databaseName = "bank";
var collectionName = "account";
var amountToTransfer = 500;

var session = db.getMongo().startSession({causalConsistency: false});
var bankDB = session.getDatabase(databaseName);
var accountColl = bankDB[collectionName];
accountColl.drop();

accountColl.insert({name: "Alice", balance: 1000});
accountColl.insert({name: "Bob", balance: 1000});

session.startTransaction();

// deduct $500 from Alice's account
var aliceBalance = accountColl.find({"name": "Alice"}).next().balance;
var newAliceBalance = aliceBalance - amountToTransfer;
accountColl.update({"name": "Alice"},{"$set": {"balance": newAliceBalance}});
var findAliceBalance = accountColl.find({"name": "Alice"}).next().balance;

// add $500 to Bob's account
var bobBalance = accountColl.find({"name": "Bob"}).next().balance;
var newBobBalance = bobBalance + amountToTransfer;
accountColl.update({"name": "Bob"},{"$set": {"balance": newBobBalance}});
var findBobBalance = accountColl.find({"name": "Bob"}).next().balance;

session.commitTransaction();

accountColl.find();

// *** Transfer $500 from Alice to Bob inside a transaction: Failure Scenario***

// Setup bank account for Alice and Bob. Each have $1000 in their account
var databaseName = "bank";
var collectionName = "account";
var amountToTransfer = 500;

var session = db.getMongo().startSession({causalConsistency: false});
var bankDB = session.getDatabase(databaseName);
```

```
var accountColl = bankDB[collectionName];
accountColl.drop();

accountColl.insert({name: "Alice", balance: 1000});
accountColl.insert({name: "Bob", balance: 1000});

session.startTransaction();

// deduct $500 from Alice's account
var aliceBalance = accountColl.find({"name": "Alice"}).next().balance;
var newAliceBalance = aliceBalance - amountToTransfer;
accountColl.update({"name": "Alice"},{"$set": {"balance": newAliceBalance}});
var findAliceBalance = accountColl.find({"name": "Alice"}).next().balance;

session.abortTransaction();
```

Multi-collection Transaktionen

Bei unseren Transaktionen handelt es sich ebenfalls um eine Mehrfacherhebung, was bedeutet, dass sie verwendet werden können, um mehrere Operationen innerhalb einer einzigen Transaktion und über mehrere Inkassovorgänge hinweg durchzuführen. Dies bietet einen konsistenten Überblick über die Daten und gewährleistet die Integrität Ihrer Daten. Wenn Sie die Befehle einzeln ausführen<>, handelt es sich bei den Transaktionen um Alles-oder-Nichts-Ausführungen, d. h., sie sind entweder alle erfolgreich oder alle schlagen fehl.

Im Folgenden finden Sie ein Beispiel für Transaktionen mit mehreren Datensätzen, bei denen dasselbe Szenario und dieselben Daten aus dem Beispiel für Transaktionen mit mehreren Anweisungen verwendet werden.

```
// *** Transfer $500 from Alice to Bob inside a transaction: Success Scenario***

// Setup bank account for Alice and Bob. Each have $1000 in their account
var amountToTransfer = 500;
var collectionName = "account";

var session = db.getMongo().startSession({causalConsistency: false});
var accountCollInBankA = session.getDatabase("bankA")[collectionName];
var accountCollInBankB = session.getDatabase("bankB")[collectionName];

accountCollInBankA.drop();
```

```
accountCollInBankB.drop();

accountCollInBankA.insert({name: "Alice", balance: 1000});
accountCollInBankB.insert({name: "Bob", balance: 1000});

session.startTransaction();

// deduct $500 from Alice's account
var aliceBalance = accountCollInBankA.find({"name": "Alice"}).next().balance;
var newAliceBalance = aliceBalance - amountToTransfer;
accountCollInBankA.update({"name": "Alice"},{"$set": {"balance": newAliceBalance}});
var findAliceBalance = accountCollInBankA.find({"name": "Alice"}).next().balance;

// add $500 to Bob's account
var bobBalance = accountCollInBankB.find({"name": "Bob"}).next().balance;
var newBobBalance = bobBalance + amountToTransfer;
accountCollInBankB.update({"name": "Bob"},{"$set": {"balance": newBobBalance}});
var findBobBalance = accountCollInBankB.find({"name": "Bob"}).next().balance;

session.commitTransaction();

accountCollInBankA.find(); // Alice holds $500 in bankA
accountCollInBankB.find(); // Bob holds $1500 in bankB

// *** Transfer $500 from Alice to Bob inside a transaction: Failure Scenario***

// Setup bank account for Alice and Bob. Each have $1000 in their account
var collectionName = "account";
var amountToTransfer = 500;

var session = db.getMongo().startSession({causalConsistency: false});
var accountCollInBankA = session.getDatabase("bankA")[collectionName];
var accountCollInBankB = session.getDatabase("bankB")[collectionName];

accountCollInBankA.drop();
accountCollInBankB.drop();

accountCollInBankA.insert({name: "Alice", balance: 1000});
accountCollInBankB.insert({name: "Bob", balance: 1000});

session.startTransaction();

// deduct $500 from Alice's account
var aliceBalance = accountCollInBankA.find({"name": "Alice"}).next().balance;
```

```
var newAliceBalance = aliceBalance - amountToTransfer;
accountCollInBankA.update({"name": "Alice"},{"$set": {"balance": newAliceBalance}});
var findAliceBalance = accountCollInBankA.find({"name": "Alice"}).next().balance;

// add $500 to Bob's account
var bobBalance = accountCollInBankB.find({"name": "Bob"}).next().balance;
var newBobBalance = bobBalance + amountToTransfer;
accountCollInBankB.update({"name": "Bob"},{"$set": {"balance": newBobBalance}});
var findBobBalance = accountCollInBankB.find({"name": "Bob"}).next().balance;

session.abortTransaction();

accountCollInBankA.find(); // Alice holds $1000 in bankA
accountCollInBankB.find(); // Bob holds $1000 in bankB
```

Beispiele für Transaktions-APIs für die Callback-API

Die Callback-API ist nur für Treiber ab Version 4.2 verfügbar.

Javascript

Der folgende Code zeigt, wie Sie die Amazon DocumentDB-Transaktions-API mit Javascript verwenden.

```
// *** Transfer $500 from Alice to Bob inside a transaction: Success ***
// Setup bank account for Alice and Bob. Each have $1000 in their account
var databaseName = "bank";
var collectionName = "account";
var amountToTransfer = 500;

var session = db.getMongo().startSession({causalConsistency: false});
var bankDB = session.getDatabase(databaseName);
var accountColl = bankDB[collectionName];
accountColl.drop();

accountColl.insert({name: "Alice", balance: 1000});
accountColl.insert({name: "Bob", balance: 1000});

session.startTransaction();

// deduct $500 from Alice's account
var aliceBalance = accountColl.find({"name": "Alice"}).next().balance;
```

```
assert(aliceBalance >= amountToTransfer);
var newAliceBalance = aliceBalance - amountToTransfer;
accountColl.update({"name": "Alice"},{"$set": {"balance": newAliceBalance}});
var findAliceBalance = accountColl.find({"name": "Alice"}).next().balance;
assert.eq(newAliceBalance, findAliceBalance);

// add $500 to Bob's account
var bobBalance = accountColl.find({"name": "Bob"}).next().balance;
var newBobBalance = bobBalance + amountToTransfer;
accountColl.update({"name": "Bob"},{"$set": {"balance": newBobBalance}});
var findBobBalance = accountColl.find({"name": "Bob"}).next().balance;
assert.eq(newBobBalance, findBobBalance);

session.commitTransaction();

accountColl.find();
```

Node.js

Der folgende Code zeigt, wie Sie die Amazon DocumentDB-Transaktions-API mit verwenden. Node.js

```
// Node.js callback API:

const bankDB = await mongoclient.db("bank");
var accountColl = await bankDB.createCollection("account");
var amountToTransfer = 500;

const session = mongoclient.startSession({causalConsistency: false});
await accountColl.drop();

await accountColl.insertOne({name: "Alice", balance: 1000}, { session });
await accountColl.insertOne({name: "Bob", balance: 1000}, { session });

const transactionOptions = {
    readConcern: { level: 'snapshot' },
    writeConcern: { w: 'majority' }
    };

// deduct $500 from Alice's account
var aliceBalance = await accountColl.findOne({name: "Alice"}, {session});
assert(aliceBalance.balance >= amountToTransfer);
var newAliceBalance = aliceBalance - amountToTransfer;
```

```
session.startTransaction(transactionOptions);
await accountColl.updateOne({name: "Alice"}, {$set: {balance: newAliceBalance}},
 {session });
await session.commitTransaction();
aliceBalance = await accountColl.findOne({name: "Alice"}, {session});
assert(newAliceBalance == aliceBalance.balance);

// add $500 to Bob's account
var bobBalance = await accountColl.findOne({name: "Bob"}, {session});
var newBobBalance = bobBalance.balance + amountToTransfer;
session.startTransaction(transactionOptions);
await accountColl.updateOne({name: "Bob"}, {$set: {balance: newBobBalance}},
 {session });
await session.commitTransaction();
bobBalance = await accountColl.findOne({name: "Bob"}, {session});
assert(newBobBalance == bobBalance.balance);
```

C#

Der folgende Code zeigt, wie die Amazon DocumentDB-Transaktions-API mit C# verwendet wird.

```
// C# Callback API

var dbName = "bank";
var collName = "account";
var amountToTransfer = 500;

using (var session = client.StartSession(new ClientSessionOptions{CausalConsistency
 = false}))
{
    var bankDB = client.GetDatabase(dbName);
    var accountColl = bankDB.GetCollection<BsonDocument>(collName);
    bankDB.DropCollection(collName);
    accountColl.InsertOne(session, new BsonDocument { {"name", "Alice"}, {"balance",
 1000 } });
    accountColl.InsertOne(session, new BsonDocument { {"name", "Bob"}, {"balance",
 1000 } });

    // start transaction
    var transactionOptions = new TransactionOptions(
            readConcern: ReadConcern.Snapshot,
            writeConcern: WriteConcern.WMajority);
    var result = session.WithTransaction(
        (sess, cancellationtoken) =>
```

```
        {
            // deduct $500 from Alice's account
            var aliceBalance = accountColl.Find(sess,
 Builders<BsonDocument>.Filter.Eq("name",
 "Alice")).FirstOrDefault().GetValue("balance");
            Debug.Assert(aliceBalance >= amountToTransfer);
            var newAliceBalance = aliceBalance.AsInt32 - amountToTransfer;
            accountColl.UpdateOne(sess, Builders<BsonDocument>.Filter.Eq("name",
 "Alice"),
                                    Builders<BsonDocument>.Update.Set("balance",
 newAliceBalance));
            aliceBalance = accountColl.Find(sess,
 Builders<BsonDocument>.Filter.Eq("name",
 "Alice")).FirstOrDefault().GetValue("balance");
            Debug.Assert(aliceBalance == newAliceBalance);

            // add $500 from Bob's account
            var bobBalance = accountColl.Find(sess,
 Builders<BsonDocument>.Filter.Eq("name",
 "Bob")).FirstOrDefault().GetValue("balance");
            var newBobBalance = bobBalance.AsInt32 + amountToTransfer;
            accountColl.UpdateOne(sess, Builders<BsonDocument>.Filter.Eq("name",
 "Bob"),
                                    Builders<BsonDocument>.Update.Set("balance",
 newBobBalance));
            bobBalance = accountColl.Find(sess,
 Builders<BsonDocument>.Filter.Eq("name",
 "Bob")).FirstOrDefault().GetValue("balance");
            Debug.Assert(bobBalance == newBobBalance);

            return "Transaction committed";
        }, transactionOptions);
  // check values outside of transaction
    var aliceNewBalance = accountColl.Find(Builders<BsonDocument>.Filter.Eq("name",
 "Alice")).FirstOrDefault().GetValue("balance");
    var bobNewBalance = accountColl.Find(Builders<BsonDocument>.Filter.Eq("name",
 "Bob")).FirstOrDefault().GetValue("balance");
    Debug.Assert(aliceNewBalance ==  500);
    Debug.Assert(bobNewBalance ==  1500);
}
```

Ruby

Der folgende Code zeigt, wie die Amazon DocumentDB-Transaktions-API mit Ruby verwendet wird.

```
// Ruby Callback API

dbName = "bank"
collName = "account"
amountToTransfer = 500

session = client.start_session(:causal_consistency=> false)
bankDB = Mongo::Database.new(client, dbName)
accountColl = bankDB[collName]
accountColl.drop()

accountColl.insert_one({"name"=>"Alice", "balance"=>1000})
accountColl.insert_one({"name"=>"Bob", "balance"=>1000})

    # start transaction
    session.with_transaction(read_concern: {level: :snapshot}, write_concern:
 {w: :majority}) do
        # deduct $500 from Alice's account
        aliceBalance = accountColl.find({"name"=>"Alice"}, :session=>
 session).first['balance']
        assert aliceBalance >= amountToTransfer
        newAliceBalance = aliceBalance - amountToTransfer
        accountColl.update_one({"name"=>"Alice"}, { "$set" =>
 {"balance"=>newAliceBalance} }, :session=> session)
        aliceBalance = accountColl.find({"name"=>>"Alice"}, :session=>
 session).first['balance']
        assert_equal(newAliceBalance, aliceBalance)

        # add $500 from Bob's account
        bobBalance = accountColl.find({"name"=>"Bob"}, :session=>
 session).first['balance']
        newBobBalance = bobBalance + amountToTransfer
        accountColl.update_one({"name"=>"Bob"}, { "$set" =>
 {"balance"=>newBobBalance} }, :session=> session)
        bobBalance = accountColl.find({"name"=>"Bob"}, :session=>
 session).first['balance']
        assert_equal(newBobBalance, bobBalance)
    end
```

```
   # check results outside of transaction
    aliceBalance = accountColl.find({"name"=>"Alice"}).first['balance']
    bobBalance = accountColl.find({"name"=>"Bob"}).first['balance']
    assert_equal(aliceBalance, 500)
    assert_equal(bobBalance, 1500)

session.end_session
```

Go

Der folgende Code zeigt, wie die Amazon DocumentDB-Transaktions-API mit Go verwendet wird.

```
// Go - Callback API
type Account struct {
    Name string
    Balance  int
}

ctx := context.TODO()

dbName := "bank"
collName := "account"
amountToTransfer := 500

session, err := client.StartSession(options.Session().SetCausalConsistency(false))
assert.NilError(t, err)
defer session.EndSession(ctx)

bankDB := client.Database(dbName)
accountColl := bankDB.Collection(collName)
accountColl.Drop(ctx)

_, err = accountColl.InsertOne(ctx, bson.M{"name" : "Alice", "balance":1000})
_, err = accountColl.InsertOne(ctx, bson.M{"name" : "Bob", "balance":1000})

transactionOptions := options.Transaction().SetReadConcern(readconcern.Snapshot()).

 SetWriteConcern(writeconcern.New(writeconcern.WMajority()))
_, err = session.WithTransaction(ctx, func(sessionCtx mongo.SessionContext)
 (interface{}, error) {
    var result Account
    // deduct $500 from Alice's account
    err = accountColl.FindOne(sessionCtx, bson.M{"name": "Alice"}).Decode(&result)
    aliceBalance := result.Balance
```

```
    newAliceBalance := aliceBalance - amountToTransfer
    _, err = accountColl.UpdateOne(sessionCtx, bson.M{"name": "Alice"},
 bson.M{"$set": bson.M{"balance": newAliceBalance}})
    err = accountColl.FindOne(sessionCtx, bson.M{"name": "Alice"}).Decode(&result)
    aliceBalance = result.Balance
    assert.Equal(t, aliceBalance, newAliceBalance)

    // add $500 to Bob's account
    err = accountColl.FindOne(sessionCtx, bson.M{"name": "Bob"}).Decode(&result)
    bobBalance := result.Balance
    newBobBalance := bobBalance + amountToTransfer
    _, err = accountColl.UpdateOne(sessionCtx, bson.M{"name": "Bob"}, bson.M{"$set":
 bson.M{"balance": newBobBalance}})
    err = accountColl.FindOne(sessionCtx, bson.M{"name": "Bob"}).Decode(&result)
    bobBalance = result.Balance
    assert.Equal(t, bobBalance, newBobBalance)

    if err != nil {
        return nil, err
    }
    return "transaction committed", err
}, transactionOptions)

// check results outside of transaction
var result Account
err = accountColl.FindOne(ctx, bson.M{"name": "Alice"}).Decode(&result)
aliceNewBalance := result.Balance
err = accountColl.FindOne(ctx, bson.M{"name": "Bob"}).Decode(&result)
bobNewBalance := result.Balance
assert.Equal(t, aliceNewBalance, 500)
assert.Equal(t, bobNewBalance, 1500)
```

Java

Der folgende Code zeigt, wie die Amazon DocumentDB-Transaktions-API mit Java verwendet wird.

```
// Java (sync) - Callback API
MongoDatabase bankDB = mongoClient.getDatabase("bank");
MongoCollection accountColl = bankDB.getCollection("account");
accountColl.drop();
int amountToTransfer = 500;
```

```
// add sample data
accountColl.insertOne(new Document("name", "Alice").append("balance", 1000));
accountColl.insertOne(new Document("name", "Bob").append("balance", 1000));

TransactionOptions txnOptions = TransactionOptions.builder()
        .readConcern(ReadConcern.SNAPSHOT)
        .writeConcern(WriteConcern.MAJORITY)
        .build();
ClientSessionOptions sessionOptions =
 ClientSessionOptions.builder().causallyConsistent(false).build();
try ( ClientSession clientSession = mongoClient.startSession(sessionOptions) ) {
    clientSession.withTransaction(new TransactionBody<Void>() {
        @Override
        public Void execute() {
            // deduct $500 from Alice's account
            List<Document> documentList = new ArrayList<>();
            accountColl.find(clientSession, new Document("name",
 "Alice")).into(documentList);
            int aliceBalance = (int) documentList.get(0).get("balance");
            int newAliceBalance = aliceBalance - amountToTransfer;

            accountColl.updateOne(clientSession, new Document("name", "Alice"), new
 Document("$set", new Document("balance", newAliceBalance)));

            // check Alice's new balance
            documentList = new ArrayList<>();
            accountColl.find(clientSession, new Document("name",
 "Alice")).into(documentList);
            int updatedBalance = (int) documentList.get(0).get("balance");
            Assert.assertEquals(updatedBalance, newAliceBalance);

            // add $500 to Bob's account
            documentList = new ArrayList<>();
            accountColl.find(clientSession, new Document("name",
 "Bob")).into(documentList);
            int bobBalance = (int) documentList.get(0).get("balance");
            int newBobBalance = bobBalance + amountToTransfer;

            accountColl.updateOne(clientSession, new Document("name", "Bob"), new
 Document("$set", new Document("balance", newBobBalance)));

            // check Bob's new balance
            documentList = new ArrayList<>();
```

```
            accountColl.find(clientSession, new Document("name",
 "Bob")).into(documentList);
            updatedBalance = (int) documentList.get(0).get("balance");
            Assert.assertEquals(updatedBalance, newBobBalance);

            return null;
        }
    }, txnOptions);
}
```

C

Der folgende Code zeigt, wie die Amazon DocumentDB-Transaktions-API mit C verwendet wird.

```
// Sample Code for C with Callback

#include <bson.h>
#include <mongoc.h>
#include <stdio.h>
#include <string.h>
#include <assert.h>

typedef struct {
    int64_t balance;
    bson_t *account;
    bson_t *opts;
    mongoc_collection_t *collection;
} ctx_t;

bool callback_session (mongoc_client_session_t *session, void *ctx, bson_t **reply,
 bson_error_t *error)
{
    bool r = true;
    ctx_t *data = (ctx_t *) ctx;
    bson_t local_reply;
    bson_t *selector = data->account;
    bson_t *update = BCON_NEW ("$set", "{", "balance", BCON_INT64 (data->balance),
 "}");

    mongoc_collection_update_one (data->collection, selector, update, data->opts,
 &local_reply, error);

    *reply = bson_copy (&local_reply);
    bson_destroy (&local_reply);
```

```
    bson_destroy (update);
    return r;
}

void test_callback_money_transfer(mongoc_client_t* client, mongoc_collection_t*
 collection, int amount_to_transfer){

    bson_t reply;
    bool r = true;
    const bson_t *doc;
    bson_iter_t iter;
    ctx_t alice_ctx;
    ctx_t bob_ctx;
    bson_error_t error;

    // find query
    bson_t *alice_query = bson_new ();
    BSON_APPEND_UTF8(alice_query, "name", "Alice");

    bson_t *bob_query = bson_new ();
    BSON_APPEND_UTF8(bob_query, "name", "Bob");

    // create session
    // set causal consistency to false
    mongoc_session_opt_t *session_opts = mongoc_session_opts_new ();
    mongoc_session_opts_set_causal_consistency (session_opts, false);
    // start the session
    mongoc_client_session_t *client_session = mongoc_client_start_session (client,
 session_opts, &error);

    // add session to options
    bson_t *opts = bson_new();
    mongoc_client_session_append (client_session, opts, &error);

    // deduct 500 from Alice
    // find account balance of Alice
    mongoc_cursor_t *cursor = mongoc_collection_find_with_opts (collection,
 alice_query, NULL, NULL);
    mongoc_cursor_next (cursor, &doc);
    bson_iter_init (&iter, doc);
    bson_iter_find (&iter, "balance");
    int64_t alice_balance = (bson_iter_value (&iter))->value.v_int64;
    assert(alice_balance >= amount_to_transfer);
    int64_t new_alice_balance = alice_balance - amount_to_transfer;
```

```
   // set variables which will be used by callback function
   alice_ctx.collection = collection;
   alice_ctx.opts = opts;
   alice_ctx.balance = new_alice_balance;
   alice_ctx.account = alice_query;

   // callback
   r = mongoc_client_session_with_transaction (client_session, &callback_session,
NULL, &alice_ctx, &reply, &error);
   assert(r);

   // find account balance of Alice after transaction
   cursor = mongoc_collection_find_with_opts (collection, alice_query, NULL, NULL);
   mongoc_cursor_next (cursor, &doc);
   bson_iter_init (&iter, doc);
   bson_iter_find (&iter, "balance");
   alice_balance = (bson_iter_value (&iter))->value.v_int64;
   assert(alice_balance == new_alice_balance);
   assert(alice_balance == 500);

       // add 500 to bob's balance
   // find account balance of Bob
   cursor = mongoc_collection_find_with_opts (collection, bob_query, NULL, NULL);
   mongoc_cursor_next (cursor, &doc);
   bson_iter_init (&iter, doc);
   bson_iter_find (&iter, "balance");
   int64_t bob_balance = (bson_iter_value (&iter))->value.v_int64;
   int64_t new_bob_balance = bob_balance + amount_to_transfer;

   bob_ctx.collection = collection;
   bob_ctx.opts = opts;
   bob_ctx.balance = new_bob_balance;
   bob_ctx.account = bob_query;

   // set read & write concern
   mongoc_read_concern_t *read_concern = mongoc_read_concern_new ();
   mongoc_write_concern_t *write_concern = mongoc_write_concern_new ();
   mongoc_transaction_opt_t *txn_opts = mongoc_transaction_opts_new ();

   mongoc_write_concern_set_w(write_concern, MONGOC_WRITE_CONCERN_W_MAJORITY);
   mongoc_read_concern_set_level(read_concern, MONGOC_READ_CONCERN_LEVEL_SNAPSHOT);
   mongoc_transaction_opts_set_write_concern (txn_opts, write_concern);
   mongoc_transaction_opts_set_read_concern (txn_opts, read_concern);
```

```
    // callback
    r = mongoc_client_session_with_transaction (client_session, &callback_session,
 txn_opts, &bob_ctx, &reply, &error);
    assert(r);

 // find account balance of Bob after transaction
    cursor = mongoc_collection_find_with_opts (collection, bob_query, NULL, NULL);
    mongoc_cursor_next (cursor, &doc);
    bson_iter_init (&iter, doc);
    bson_iter_find (&iter, "balance");
    bob_balance = (bson_iter_value (&iter))->value.v_int64;
    assert(bob_balance == new_bob_balance);
    assert(bob_balance == 1500);

    // cleanup
    bson_destroy(alice_query);
    bson_destroy(bob_query);
    mongoc_client_session_destroy(client_session);
    bson_destroy(opts);
    mongoc_transaction_opts_destroy(txn_opts);
    mongoc_read_concern_destroy(read_concern);
    mongoc_write_concern_destroy(write_concern);
    mongoc_cursor_destroy(cursor);
    bson_destroy(doc);
}
int main(int argc, char* argv[]) {
    mongoc_init ();
    mongoc_client_t* client = mongoc_client_new (<connection uri>);
    bson_error_t error;

    // connect to bank db
    mongoc_database_t *database = mongoc_client_get_database (client, "bank");
    // access account collection
    mongoc_collection_t* collection = mongoc_client_get_collection(client, "bank",
 "account");
    // set amount to transfer
    int64_t amount_to_transfer = 500;
    // delete the collection if already existing
    mongoc_collection_drop(collection, &error);

    // open Alice account
    bson_t *alice_account = bson_new ();
    BSON_APPEND_UTF8(alice_account, "name", "Alice");
```

```
    BSON_APPEND_INT64(alice_account, "balance", 1000);

    // open Bob account
    bson_t *bob_account = bson_new ();
    BSON_APPEND_UTF8(bob_account, "name", "Bob");
    BSON_APPEND_INT64(bob_account, "balance", 1000);

    bool r = true;

    r = mongoc_collection_insert_one(collection, alice_account, NULL, NULL, &error);
    if (!r) {printf("Error encountered:%s", error.message);}
    r = mongoc_collection_insert_one(collection, bob_account, NULL, NULL, &error);
    if (!r) {printf("Error encountered:%s", error.message);}

    test_callback_money_transfer(client, collection, amount_to_transfer);

}
```

Python

Der folgende Code zeigt, wie die Amazon DocumentDB-Transaktions-API mit Python verwendet wird.

```
// Sample Python code with callback api

import pymongo

def callback(session, balance, query):
    collection.update_one(query, {'$set': {"balance": balance}}, session=session)

client = pymongo.MongoClient(<connection uri>)
rc_snapshot = pymongo.read_concern.ReadConcern('snapshot')
wc_majority = pymongo.write_concern.WriteConcern('majority')

# To start, drop and create an account collection and insert balances for both Alice
 and Bob
collection = client.get_database("bank").get_collection("account")
collection.drop()
collection.insert_one({"_id": 1, "name": "Alice", "balance": 1000})
collection.insert_one({"_id": 2, "name": "Bob", "balance": 1000})

amount_to_transfer = 500
```

```
# deduct 500 from Alice's account
alice_balance = collection.find_one({"name": "Alice"}).get("balance")
assert alice_balance >= amount_to_transfer
new_alice_balance = alice_balance - amount_to_transfer

with client.start_session({'causalConsistency':False}) as session:
    session.with_transaction(lambda s: callback(s, new_alice_balance, {"name":
 "Alice"}), read_concern=rc_snapshot, write_concern=wc_majority)

updated_alice_balance = collection.find_one({"name": "Alice"}).get("balance")
assert updated_alice_balance == new_alice_balance

# add 500 to Bob's account
bob_balance = collection.find_one({"name": "Bob"}).get("balance")
assert bob_balance >= amount_to_transfer
new_bob_balance = bob_balance + amount_to_transfer

with client.start_session({'causalConsistency':False}) as session:
    session.with_transaction(lambda s: callback(s, new_bob_balance, {"name":
 "Bob"}), read_concern=rc_snapshot, write_concern=wc_majority)

updated_bob_balance = collection.find_one({"name": "Bob"}).get("balance")
assert updated_bob_balance == new_bob_balance
```

Beispiele für Transaktions-APIs für die Kern-API

Javascript

Der folgende Code zeigt, wie Sie die Amazon DocumentDB-Transaktions-API mit Javascript verwenden.

```
// *** Transfer $500 from Alice to Bob inside a transaction: Success ***
// Setup bank account for Alice and Bob. Each have $1000 in their account
var databaseName = "bank";
var collectionName = "account";
var amountToTransfer = 500;

var session = db.getMongo().startSession({causalConsistency: false});
var bankDB = session.getDatabase(databaseName);
var accountColl = bankDB[collectionName];
accountColl.drop();
```

```
accountColl.insert({name: "Alice", balance: 1000});
accountColl.insert({name: "Bob", balance: 1000});

session.startTransaction();

// deduct $500 from Alice's account
var aliceBalance = accountColl.find({"name": "Alice"}).next().balance;
assert(aliceBalance >= amountToTransfer);
var newAliceBalance = aliceBalance - amountToTransfer;
accountColl.update({"name": "Alice"},{"$set": {"balance": newAliceBalance}});
var findAliceBalance = accountColl.find({"name": "Alice"}).next().balance;
assert.eq(newAliceBalance, findAliceBalance);

// add $500 to Bob's account
var bobBalance = accountColl.find({"name": "Bob"}).next().balance;
var newBobBalance = bobBalance + amountToTransfer;
accountColl.update({"name": "Bob"},{"$set": {"balance": newBobBalance}});
var findBobBalance = accountColl.find({"name": "Bob"}).next().balance;
assert.eq(newBobBalance, findBobBalance);

session.commitTransaction();

accountColl.find();
```

C#

Der folgende Code zeigt, wie die Amazon DocumentDB-Transaktions-API mit C# verwendet wird.

```
// C# Core API

public void TransferMoneyWithRetry(IMongoCollection<bSondocument> accountColl,
 IClientSessionHandle session)
{
    var amountToTransfer = 500;

    // start transaction
   var transactionOptions = new TransactionOptions(
                readConcern: ReadConcern.Snapshot,
                writeConcern: WriteConcern.WMajority);
    session.StartTransaction(transactionOptions);
   try
    {
        // deduct $500 from Alice's account
```

```
        var aliceBalance = accountColl.Find(session,
 Builders<bSondocument>.Filter.Eq("name",
 "Alice")).FirstOrDefault().GetValue("balance");
        Debug.Assert(aliceBalance >= amountToTransfer);
        var newAliceBalance = aliceBalance.AsInt32 - amountToTransfer;
        accountColl.UpdateOne(session, Builders<bSondocument>.Filter.Eq("name",
 "Alice"),
                                Builders<bSondocument>.Update.Set("balance",
 newAliceBalance));
        aliceBalance = accountColl.Find(session,
 Builders<bSondocument>.Filter.Eq("name",
 "Alice")).FirstOrDefault().GetValue("balance");
        Debug.Assert(aliceBalance == newAliceBalance);

        // add $500 from Bob's account
        var bobBalance = accountColl.Find(session,
 Builders<bSondocument>.Filter.Eq("name",
 "Bob")).FirstOrDefault().GetValue("balance");
        var newBobBalance = bobBalance.AsInt32 + amountToTransfer;
        accountColl.UpdateOne(session, Builders<bSondocument>.Filter.Eq("name",
 "Bob"),
                                Builders<bSondocument>.Update.Set("balance",
 newBobBalance));
        bobBalance = accountColl.Find(session,
 Builders<bSondocument>.Filter.Eq("name",
 "Bob")).FirstOrDefault().GetValue("balance");
        Debug.Assert(bobBalance == newBobBalance);

    }
    catch (Exception e)
    {
        session.AbortTransaction();
        throw;
    }

    session.CommitTransaction();
 }

}
public void DoTransactionWithRetry(MongoClient client)
{
    var dbName = "bank";
    var collName = "account";
```

```
   using (var session = client.StartSession(new
ClientSessionOptions{CausalConsistency = false}))
   {
       try
       {
           var bankDB = client.GetDatabase(dbName);
           var accountColl = bankDB.GetCollection<bSondocument>(collName);
           bankDB.DropCollection(collName);
           accountColl.InsertOne(session, new BsonDocument { {"name", "Alice"},
{"balance", 1000 } });
           accountColl.InsertOne(session, new BsonDocument { {"name", "Bob"},
{"balance", 1000 } });

           while(true) {
               try
               {
                       TransferMoneyWithRetry(accountColl, session);
                       break;
               }
               catch (MongoException e)
               {
                   if(e.HasErrorLabel("TransientTransactionError"))
                   {
                       continue;
                   }
                   else
                   {
                       throw;
                   }
               }
           }

           // check values outside of transaction
           var aliceNewBalance =
accountColl.Find(Builders<bSondocument>.Filter.Eq("name",
"Alice")).FirstOrDefault().GetValue("balance");
           var bobNewBalance =
accountColl.Find(Builders<bSondocument>.Filter.Eq("name",
"Bob")).FirstOrDefault().GetValue("balance");
           Debug.Assert(aliceNewBalance ==  500);
           Debug.Assert(bobNewBalance ==  1500);
       }
       catch (Exception e)
       {
```

```
            Console.WriteLine("Error running transaction: " + e.Message);
        }
    }
}
```

Ruby

Der folgende Code zeigt, wie die Amazon DocumentDB-Transaktions-API mit Ruby verwendet wird.

```
# Ruby Core API

def transfer_money_w_retry(session, accountColl)
    amountToTransfer = 500

    session.start_transaction(read_concern: {level: :snapshot}, write_concern:
 {w: :majority})
    # deduct $500 from Alice's account
    aliceBalance = accountColl.find({"name"=>"Alice"}, :session=>
 session).first['balance']
    assert aliceBalance >= amountToTransfer
    newAliceBalance = aliceBalance - amountToTransfer
    accountColl.update_one({"name"=>"Alice"}, { "$set" =>
 {"balance"=>newAliceBalance} }, :session=> session)
    aliceBalance = accountColl.find({"name"=>"Alice"}, :session=>
 session).first['balance']
    assert_equal(newAliceBalance, aliceBalance)

    # add $500 to Bob's account
    bobBalance = accountColl.find({"name"=>"Bob"}, :session=>
 session).first['balance']
    newBobBalance = bobBalance + amountToTransfer
    accountColl.update_one({"name"=>"Bob"}, { "$set" =>
 {"balance"=>newBobBalance} }, :session=> session)
    bobBalance = accountColl.find({"name"=>"Bob"}, :session=>
 session).first['balance']
    assert_equal(newBobBalance, bobBalance)

    session.commit_transaction

end

def do_txn_w_retry(client)
```

```
     dbName = "bank"
    collName = "account"

    session = client.start_session(:causal_consistency=> false)
    bankDB = Mongo::Database.new(client, dbName)
    accountColl = bankDB[collName]
    accountColl.drop()

    accountColl.insert_one({"name"=>"Alice", "balance"=>1000})
    accountColl.insert_one({"name"=>"Bob", "balance"=>1000})

    begin
        transferMoneyWithRetry(session, accountColl)
        puts "transaction committed"
    rescue Mongo::Error => e
        if e.label?('TransientTransactionError')
            retry
        else
            puts "transaction failed"
            raise
        end
    end

    # check results outside of transaction
    aliceBalance = accountColl.find({"name"=>"Alice"}).first['balance']
    bobBalance = accountColl.find({"name"=>"Bob"}).first['balance']
    assert_equal(aliceBalance, 500)
    assert_equal(bobBalance, 1500)

end
```

Go

Der folgende Code zeigt, wie die Amazon DocumentDB-Transaktions-API mit Go verwendet wird.

```
// Go - Core API
type Account struct {
    Name string
    Balance  int
}

func transferMoneyWithRetry(sessionContext mongo.SessionContext, accountColl
 *mongo.Collection, t *testing.T) error {
    amountToTransfer := 500
```

```
    transactionOptions :=
 options.Transaction().SetReadConcern(readconcern.Snapshot()).

 SetWriteConcern(writeconcern.New(writeconcern.WMajority()))
    if err := sessionContext.StartTransaction(transactionOptions); err != nil {
        panic(err)
    }

    var result Account
    // deduct $500 from Alice's account
    err := accountColl.FindOne(sessionContext, bson.M{"name":
 "Alice"}).Decode(&result)
    aliceBalance := result.Balance
    newAliceBalance := aliceBalance - amountToTransfer
    _, err = accountColl.UpdateOne(sessionContext, bson.M{"name": "Alice"},
 bson.M{"$set": bson.M{"balance": newAliceBalance}})
    if err != nil {
        sessionContext.AbortTransaction(sessionContext)
    }
    err = accountColl.FindOne(sessionContext, bson.M{"name":
 "Alice"}).Decode(&result)
    aliceBalance = result.Balance
    assert.Equal(t, aliceBalance, newAliceBalance)

    // add $500 to Bob's account
    err = accountColl.FindOne(sessionContext, bson.M{"name": "Bob"}).Decode(&result)
    bobBalance := result.Balance
    newBobBalance := bobBalance + amountToTransfer
    _, err = accountColl.UpdateOne(sessionContext, bson.M{"name": "Bob"},
 bson.M{"$set": bson.M{"balance": newBobBalance}})
    if err != nil {
        sessionContext.AbortTransaction(sessionContext)
    }
    err = accountColl.FindOne(sessionContext, bson.M{"name": "Bob"}).Decode(&result)
    bobBalance = result.Balance
    assert.Equal(t, bobBalance, newBobBalance)

    err = sessionContext.CommitTransaction(sessionContext)
    return err
}

func doTransactionWithRetry(t *testing.T) {
    ctx := context.TODO()
```

```
    dbName := "bank"
    collName := "account"
    bankDB := client.Database(dbName)
    accountColl := bankDB.Collection(collName)

    client.UseSessionWithOptions(ctx, options.Session().SetCausalConsistency(false),
 func(sessionContext mongo.SessionContext) error {
        accountColl.Drop(ctx)
        accountColl.InsertOne(sessionContext, bson.M{"name" : "Alice",
 "balance":1000})
        accountColl.InsertOne(sessionContext, bson.M{"name" : "Bob",
 "balance":1000})
        for {
            err := transferMoneyWithRetry(sessionContext, accountColl, t)
            if err == nil {
                println("transaction committed")
                return nil
            }
            if mongoErr := err.(mongo.CommandError);
 mongoErr.HasErrorLabel("TransientTransactionError") {
                continue
            }
            println("transaction failed")
            return err
        }
    })

    // check results outside of transaction
    var result Account
    accountColl.FindOne(ctx, bson.M{"name": "Alice"}).Decode(&result)
    aliceBalance := result.Balance
    assert.Equal(t, aliceBalance, 500)
    accountColl.FindOne(ctx, bson.M{"name": "Bob"}).Decode(&result)
    bobBalance := result.Balance
    assert.Equal(t, bobBalance, 1500)
}
```

Java

Der folgende Code zeigt, wie die Amazon DocumentDB-Transaktions-API mit Java verwendet wird.

```
// Java (sync) - Core API

public void transferMoneyWithRetry() {
   // connect to server
    MongoClientURI mongoURI = new MongoClientURI(uri);
    MongoClient mongoClient = new MongoClient(mongoURI);

    MongoDatabase bankDB = mongoClient.getDatabase("bank");
    MongoCollection accountColl = bankDB.getCollection("account");
    accountColl.drop();

   // insert some sample data
    accountColl.insertOne(new Document("name", "Alice").append("balance", 1000));
    accountColl.insertOne(new Document("name", "Bob").append("balance", 1000));

    while (true) {
        try {
            doTransferMoneyWithRetry(accountColl, mongoClient);
            break;
        } catch (MongoException e) {
            if (e.hasErrorLabel(MongoException.TRANSIENT_TRANSACTION_ERROR_LABEL)) {
                continue;
            } else {
                throw e;
            }
        }
    }
}

public void doTransferMoneyWithRetry(MongoCollection accountColl, MongoClient
 mongoClient) {
    int amountToTransfer = 500;

   TransactionOptions txnOptions = TransactionOptions.builder()
      .readConcern(ReadConcern.SNAPSHOT)
      .writeConcern(WriteConcern.MAJORITY)
      .build();
    ClientSessionOptions sessionOptions =
 ClientSessionOptions.builder().causallyConsistent(false).build();
    try ( ClientSession clientSession = mongoClient.startSession(sessionOptions) ) {
        clientSession.startTransaction(txnOptions);

        // deduct $500 from Alice's account
```

```
        List<Document> documentList = new ArrayList<>();
        accountColl.find(clientSession, new Document("name",
 "Alice")).into(documentList);
        int aliceBalance = (int) documentList.get(0).get("balance");
        Assert.assertTrue(aliceBalance >= amountToTransfer);
        int newAliceBalance = aliceBalance - amountToTransfer;
        accountColl.updateOne(clientSession, new Document("name", "Alice"), new
 Document("$set", new Document("balance", newAliceBalance)));

        // check Alice's new balance
        documentList = new ArrayList<>();
        accountColl.find(clientSession, new Document("name",
 "Alice")).into(documentList);
        int updatedBalance = (int) documentList.get(0).get("balance");
        Assert.assertEquals(updatedBalance, newAliceBalance);

        // add $500 to Bob's account
        documentList = new ArrayList<>();
        accountColl.find(clientSession, new Document("name",
 "Bob")).into(documentList);
        int bobBalance = (int) documentList.get(0).get("balance");
        int newBobBalance = bobBalance + amountToTransfer;
        accountColl.updateOne(clientSession, new Document("name", "Bob"), new
 Document("$set", new Document("balance", newBobBalance)));

        // check Bob's new balance
        documentList = new ArrayList<>();
        accountColl.find(clientSession, new Document("name",
 "Bob")).into(documentList);
        updatedBalance = (int) documentList.get(0).get("balance");
        Assert.assertEquals(updatedBalance, newBobBalance);

        // commit transaction
        clientSession.commitTransaction();
    }
}
// Java (async) -- Core API
public void transferMoneyWithRetry() {
    // connect to the server
    MongoClient mongoClient = MongoClients.create(uri);

    MongoDatabase bankDB = mongoClient.getDatabase("bank");
    MongoCollection accountColl = bankDB.getCollection("account");
    SubscriberLatchWrapper<Void> dropCallback = new SubscriberLatchWrapper<>();
```

```
    mongoClient.getDatabase("bank").drop().subscribe(dropCallback);
    dropCallback.await();

    // insert some sample data
    SubscriberLatchWrapper<InsertOneResult> insertionCallback = new
 SubscriberLatchWrapper<>();
    accountColl.insertOne(new Document("name", "Alice").append("balance",
 1000)).subscribe(insertionCallback);
    insertionCallback.await();

    insertionCallback = new SubscriberLatchWrapper<>();
    accountColl.insertOne(new Document("name", "Bob").append("balance",
 1000)).subscribe(insertionCallback);;
    insertionCallback.await();

    while (true) {
        try {
            doTransferMoneyWithRetry(accountColl, mongoClient);
            break;
        } catch (MongoException e) {
            if (e.hasErrorLabel(MongoException.TRANSIENT_TRANSACTION_ERROR_LABEL)) {
                continue;
            } else {
                throw e;
            }
        }
    }
}

public void doTransferMoneyWithRetry(MongoCollection accountColl, MongoClient
 mongoClient) {
    int amountToTransfer = 500;

    // start the transaction
    TransactionOptions txnOptions = TransactionOptions.builder()
            .readConcern(ReadConcern.SNAPSHOT)
            .writeConcern(WriteConcern.MAJORITY)
            .build();
    ClientSessionOptions sessionOptions =
 ClientSessionOptions.builder().causallyConsistent(false).build();

    SubscriberLatchWrapper<ClientSession> sessionCallback = new
 SubscriberLatchWrapper<>();
    mongoClient.startSession(sessionOptions).subscribe(sessionCallback);
```

```
    ClientSession session = sessionCallback.get().get(0);
    session.startTransaction(txnOptions);

    // deduct $500 from Alice's account
    SubscriberLatchWrapper<Document> findCallback = new SubscriberLatchWrapper<>();
    accountColl.find(session, new Document("name",
"Alice")).first().subscribe(findCallback);
    Document documentFound = findCallback.get().get(0);
    int aliceBalance = (int) documentFound.get("balance");
    int newAliceBalance = aliceBalance - amountToTransfer;

    SubscriberLatchWrapper<UpdateResult> updateCallback = new
SubscriberLatchWrapper<>();
    accountColl.updateOne(session, new Document("name",
"Alice"), new Document("$set", new Document("balance",
newAliceBalance))).subscribe(updateCallback);
    updateCallback.await();

    // check Alice's new balance
    findCallback = new SubscriberLatchWrapper<>();
    accountColl.find(session, new Document("name",
"Alice")).first().subscribe(findCallback);
    documentFound = findCallback.get().get(0);
    int updatedBalance = (int) documentFound.get("balance");
    Assert.assertEquals(updatedBalance, newAliceBalance);

    // add $500 to Bob's account
    findCallback = new SubscriberLatchWrapper<>();
    accountColl.find(session, new Document("name",
"Bob")).first().subscribe(findCallback);
    documentFound = findCallback.get().get(0);
    int bobBalance = (int) documentFound.get("balance");
    int newBobBalance = bobBalance + amountToTransfer;

    updateCallback = new SubscriberLatchWrapper<>();
    accountColl.updateOne(session, new Document("name", "Bob"), new Document("$set",
new Document("balance", newBobBalance))).subscribe(updateCallback);
    updateCallback.await();

    // check Bob's new balance
    findCallback = new SubscriberLatchWrapper<>();
    accountColl.find(session, new Document("name",
"Bob")).first().subscribe(findCallback);
    documentFound = findCallback.get().get(0);
```

```
    updatedBalance = (int) documentFound.get("balance");
    Assert.assertEquals(updatedBalance, newBobBalance);

    // commit the transaction
    SubscriberLatchWrapper<Void> transactionCallback = new
 SubscriberLatchWrapper<>();
    session.commitTransaction().subscribe(transactionCallback);
    transactionCallback.await();
}

public class SubscriberLatchWrapper<T> implements Subscriber<T> {

    /**
     * A Subscriber that stores the publishers results and provides a latch so can
 block on completion.
     *
     * @param <T> The publishers result type
     */
    private final List<T> received;
    private final List<RuntimeException> errors;
    private final CountDownLatch latch;
    private volatile Subscription subscription;
    private volatile boolean completed;

    /**
     * Construct an instance
     */
    public SubscriberLatchWrapper() {
        this.received = new ArrayList<>();
        this.errors = new ArrayList<>();
        this.latch = new CountDownLatch(1);
    }

    @Override
    public void onSubscribe(final Subscription s) {
        subscription = s;
        subscription.request(Integer.MAX_VALUE);
    }

    @Override
    public void onNext(final T t) {
        received.add(t);
    }
```

```
    @Override
    public void onError(final Throwable t) {
        if (t instanceof RuntimeException) {
            errors.add((RuntimeException) t);
        } else {
            errors.add(new RuntimeException("Unexpected exception", t));
        }
        onComplete();
    }

    @Override
    public void onComplete() {
        completed = true;
        subscription.cancel();
        latch.countDown();
    }

    /**
     * Get received elements
     *
     * @return the list of received elements
     */
    public List<T> getReceived() {
        return received;
    }

    /**
     * Get received elements.
     *
     * @return the list of receive elements
     */
    public List<T> get() {
        return await().getReceived();
    }

    /**
     * Await completion or error
     *
     * @return this
     */
    public SubscriberLatchWrapper<T> await() {
        subscription.request(Integer.MAX_VALUE);
        try {
            if (!latch.await(300, TimeUnit.SECONDS)) {
```

```
                throw new MongoTimeoutException("Publisher onComplete timed out for
 300 seconds");
            }
        } catch (InterruptedException e) {
            throw new MongoInterruptedException("Interrupted waiting for
 observeration", e);
        }
        if (!errors.isEmpty()) {
            throw errors.get(0);
        }
        return this;
    }

    public boolean getCompleted() {
        return this.completed;
    }

    public void close() {
        subscription.cancel();
        received.clear();
    }
}
```

C

Der folgende Code zeigt, wie die Amazon DocumentDB-Transaktions-API mit C verwendet wird.

```
// Sample C code with core session

bool core_session(mongoc_client_session_t *client_session, mongoc_collection_t*
 collection, bson_t *selector, int64_t balance){
    bool r = true;
    bson_error_t error;
    bson_t *opts = bson_new();
    bson_t *update = BCON_NEW ("$set", "{", "balance", BCON_INT64 (balance), "}");

    // set read & write concern
    mongoc_read_concern_t *read_concern = mongoc_read_concern_new ();
    mongoc_write_concern_t *write_concern = mongoc_write_concern_new ();
    mongoc_transaction_opt_t *txn_opts = mongoc_transaction_opts_new ();

    mongoc_write_concern_set_w(write_concern, MONGOC_WRITE_CONCERN_W_MAJORITY);
    mongoc_read_concern_set_level(read_concern, MONGOC_READ_CONCERN_LEVEL_SNAPSHOT);
```

```
    mongoc_transaction_opts_set_write_concern (txn_opts, write_concern);
    mongoc_transaction_opts_set_read_concern (txn_opts, read_concern);

    mongoc_client_session_start_transaction (client_session, txn_opts, &error);
    mongoc_client_session_append (client_session, opts, &error);

    r = mongoc_collection_update_one (collection, selector, update, opts, NULL,
 &error);

    mongoc_client_session_commit_transaction (client_session, NULL, &error);
    bson_destroy (opts);
    mongoc_transaction_opts_destroy(txn_opts);
    mongoc_read_concern_destroy(read_concern);
    mongoc_write_concern_destroy(write_concern);
    bson_destroy (update);
    return r;
}

void test_core_money_transfer(mongoc_client_t* client, mongoc_collection_t*
 collection, int amount_to_transfer){

    bson_t reply;
    bool r = true;
    const bson_t *doc;
    bson_iter_t iter;
    bson_error_t error;

    // find query
    bson_t *alice_query = bson_new ();
    BSON_APPEND_UTF8(alice_query, "name", "Alice");

    bson_t *bob_query = bson_new ();
    BSON_APPEND_UTF8(bob_query, "name", "Bob");

    // create session
    // set causal consistency to false
    mongoc_session_opt_t *session_opts = mongoc_session_opts_new ();
    mongoc_session_opts_set_causal_consistency (session_opts, false);
    // start the session
    mongoc_client_session_t *client_session = mongoc_client_start_session (client,
 session_opts, &error);

    // add session to options
    bson_t *opts = bson_new();
```

```
    mongoc_client_session_append (client_session, opts, &error);

    // deduct 500 from Alice
    // find account balance of Alice
    mongoc_cursor_t *cursor = mongoc_collection_find_with_opts (collection,
alice_query, NULL, NULL);
    mongoc_cursor_next (cursor, &doc);
    bson_iter_init (&iter, doc);
    bson_iter_find (&iter, "balance");
    int64_t alice_balance = (bson_iter_value (&iter))->value.v_int64;
    assert(alice_balance >= amount_to_transfer);
    int64_t new_alice_balance = alice_balance - amount_to_transfer;

    // core
    r = core_session (client_session, collection, alice_query, new_alice_balance);
    assert(r);

    // find account balance of Alice after transaction
    cursor = mongoc_collection_find_with_opts (collection, alice_query, NULL, NULL);
    mongoc_cursor_next (cursor, &doc);
    bson_iter_init (&iter, doc);
    bson_iter_find (&iter, "balance");
    alice_balance = (bson_iter_value (&iter))->value.v_int64;
    assert(alice_balance == new_alice_balance);
    assert(alice_balance == 500);

    // add 500 to Bob's balance
    // find account balance of Bob
    cursor = mongoc_collection_find_with_opts (collection, bob_query, NULL, NULL);
    mongoc_cursor_next (cursor, &doc);
    bson_iter_init (&iter, doc);
    bson_iter_find (&iter, "balance");
    int64_t bob_balance = (bson_iter_value (&iter))->value.v_int64;
    int64_t new_bob_balance = bob_balance + amount_to_transfer;

    //core
    r = core_session (client_session, collection, bob_query, new_bob_balance);
    assert(r);

    // find account balance of Bob after transaction
    cursor = mongoc_collection_find_with_opts (collection, bob_query, NULL, NULL);
    mongoc_cursor_next (cursor, &doc);
    bson_iter_init (&iter, doc);
    bson_iter_find (&iter, "balance");
```

```
    bob_balance = (bson_iter_value (&iter))->value.v_int64;
    assert(bob_balance == new_bob_balance);
    assert(bob_balance == 1500);

    // cleanup
    bson_destroy(alice_query);
    bson_destroy(bob_query);
    mongoc_client_session_destroy(client_session);
    bson_destroy(opts);
    mongoc_cursor_destroy(cursor);
    bson_destroy(doc);
}

int main(int argc, char* argv[]) {
    mongoc_init ();
    mongoc_client_t* client = mongoc_client_new (<connection uri>);
    bson_error_t error;

    // connect to bank db
    mongoc_database_t *database = mongoc_client_get_database (client, "bank");
    // access account collection
    mongoc_collection_t* collection = mongoc_client_get_collection(client, "bank",
 "account");
    // set amount to transfer
    int64_t amount_to_transfer = 500;
    // delete the collection if already existing
    mongoc_collection_drop(collection, &error);

    // open Alice account
    bson_t *alice_account = bson_new ();
    BSON_APPEND_UTF8(alice_account, "name", "Alice");
    BSON_APPEND_INT64(alice_account, "balance", 1000);

    // open Bob account
    bson_t *bob_account = bson_new ();
    BSON_APPEND_UTF8(bob_account, "name", "Bob");
    BSON_APPEND_INT64(bob_account, "balance", 1000);

    bool r = true;

    r = mongoc_collection_insert_one(collection, alice_account, NULL, NULL, &error);
    if (!r) {printf("Error encountered:%s", error.message);}
    r = mongoc_collection_insert_one(collection, bob_account, NULL, NULL, &error);
    if (!r) {printf("Error encountered:%s", error.message);}
```

```
    test_core_money_transfer(client, collection, amount_to_transfer);

}
```

Scala

Der folgende Code zeigt, wie die Amazon DocumentDB-Transaktions-API mit Scala verwendet wird.

```
// Scala Core API
def transferMoneyWithRetry(sessionObservable: SingleObservable[ClientSession] ,
 database: MongoDatabase ): Unit = {
    val accountColl = database.getCollection("account")
    var amountToTransfer = 500

    var transactionObservable: Observable[ClientSession] =
 sessionObservable.map(clientSession => {
    clientSession.startTransaction()

    // deduct $500 from Alice's account
    var aliceBalance = accountColl.find(clientSession, Document("name" ->
 "Alice")).await().head.getInteger("balance")
    assert(aliceBalance >= amountToTransfer)
    var newAliceBalance = aliceBalance - amountToTransfer
    accountColl.updateOne(clientSession, Document("name" -> "Alice"),
 Document("$set" -> Document("balance" -> newAliceBalance))).await()
    aliceBalance = accountColl.find(clientSession, Document("name" ->
 "Alice")).await().head.getInteger("balance")
    assert(aliceBalance == newAliceBalance)

    // add $500 to Bob's account
    var bobBalance = accountColl.find(clientSession, Document("name" ->
 "Bob")).await().head.getInteger("balance")
    var newBobBalance = bobBalance + amountToTransfer
    accountColl.updateOne(clientSession, Document("name" -> "Bob"), Document("$set"
 -> Document("balance" -> newBobBalance))).await()
    bobBalance = accountColl.find(clientSession, Document("name" ->
 "Bob")).await().head.getInteger("balance")
    assert(bobBalance == newBobBalance)

    clientSession
    })
```

```
    transactionObservable.flatMap(clientSession =>
 clientSession.commitTransaction()).await()
}

def doTransactionWithRetry(): Unit = {
    val client: MongoClient = MongoClientWrapper.getMongoClient()
    val database: MongoDatabase = client.getDatabase("bank")
    val accountColl = database.getCollection("account")
    accountColl.drop().await()

    val sessionOptions =
 ClientSessionOptions.builder().causallyConsistent(false).build()
    var  sessionObservable: SingleObservable[ClientSession] =
 client.startSession(sessionOptions)
    accountColl.insertOne(Document("name" -> "Alice", "balance" -> 1000)).await()
    accountColl.insertOne(Document("name" -> "Bob", "balance" -> 1000)).await()

    var retry = true
    while (retry) {
        try {
        transferMoneyWithRetry(sessionObservable, database)
        println("transaction committed")
        retry = false
        }
        catch {
        case e: MongoException if
 e.hasErrorLabel(MongoException.TRANSIENT_TRANSACTION_ERROR_LABEL) => {
            println("retrying transaction")
        }
        case other: Throwable => {
            println("transaction failed")
            retry = false
            throw other

        }
        }
    }

    // check results outside of transaction
    assert(accountColl.find(Document("name" ->
 "Alice")).results().head.getInteger("balance") == 500)
    assert(accountColl.find(Document("name" ->
 "Bob")).results().head.getInteger("balance") == 1500)
```

```
    accountColl.drop().await()

}
```

Python

Der folgende Code zeigt, wie die Amazon DocumentDB-Transaktions-API mit Python verwendet wird.

```
// Sample Python code with Core api

import pymongo

client = pymongo.MongoClient(<connection_string>)
rc_snapshot = pymongo.read_concern.ReadConcern('snapshot')
wc_majority = pymongo.write_concern.WriteConcern('majority')

# To start, drop and create an account collection and insert balances for both Alice
 and Bob
collection = client.get_database("bank").get_collection("account")
collection.drop()
collection.insert_one({"_id": 1, "name": "Alice", "balance": 1000})
collection.insert_one({"_id": 2, "name": "Bob", "balance": 1000})

amount_to_transfer = 500

# deduct 500 from Alice's account
alice_balance = collection.find_one({"name": "Alice"}).get("balance")
assert alice_balance >= amount_to_transfer
new_alice_balance = alice_balance - amount_to_transfer

with client.start_session({'causalConsistency':False}) as session:
    session.start_transaction(read_concern=rc_snapshot, write_concern=wc_majority)
    collection.update_one({"name": "Alice"}, {'$set': {"balance":
 new_alice_balance}}, session=session)
    session.commit_transaction()

updated_alice_balance = collection.find_one({"name": "Alice"}).get("balance")
assert updated_alice_balance == new_alice_balance

# add 500 to Bob's account
bob_balance = collection.find_one({"name": "Bob"}).get("balance")
```

```
assert bob_balance >= amount_to_transfer
new_bob_balance = bob_balance + amount_to_transfer

with client.start_session({'causalConsistency':False}) as session:
    session.start_transaction(read_concern=rc_snapshot, write_concern=wc_majority)
    collection.update_one({"name": "Bob"}, {'$set': {"balance": new_bob_balance}},
 session=session)
    session.commit_transaction()

updated_bob_balance = collection.find_one({"name": "Bob"}).get("balance")
assert updated_bob_balance == new_bob_balance
```

Unterstützte Befehle

Befehl	Unterstützt
`abortTransaction`	Ja
`commitTransaction`	Ja
`endSessions`	Ja
`killSession`	Ja
`killAllSession`	Ja
`killAllSessionsByPattern`	Nein
`refreshSessions`	Nein
`startSession`	Ja

Nicht unterstützte Funktionen

Methoden	Stufen oder Befehle
`db.collection.aggregate()`	`$collStats` `$currentOp`

Methoden	Stufen oder Befehle

	`$indexStats`
	
	`$listSessions`
	
	`$out`

`db.collection.count()`
 	
`db.collection.countDocuments()`
	`$where`
	
	`$near`
	
	`$nearSphere`

`db.collection.insert()`	`insert`wird nicht unterstützt, wenn sie nicht für eine vorhandene Sammlung ausgeführt wird. Diese Methode wird unterstützt, wenn sie auf eine bereits bestehende Sammlung abzielt.

Sitzungen

MongoDB-Sitzungen sind ein Framework, das zur Unterstützung wiederholbarer Schreibversuche, kausaler Konsistenz, Transaktionen und zur datenbankübergreifenden Verwaltung von Vorgängen verwendet wird. Wenn eine Sitzung erstellt wird, wird vom Client eine logische Sitzungs-ID (lsid) generiert, die verwendet wird, um beim Senden von Befehlen an den Server alle Operationen innerhalb dieser Sitzung zu kennzeichnen.

Amazon DocumentDB unterstützt die Verwendung von Sitzungen zur Aktivierung von Transaktionen, unterstützt jedoch keine kausale Konsistenz. Ab der Engine-Version 8.0.2 unterstützt Amazon DocumentDB für Schreibvorgänge, bei denen sich jeder einzelne Schreibvorgang auf höchstens ein Dokument auswirkt, außerhalb von Transaktionen wiederholbare Schreibversuche. In früheren Engine-Versionen werden wiederholbare Schreibversuche nicht unterstützt. Weitere Informationen finden Sie unter Wiederholbare Schreibvorgänge in Amazon DocumentDB.

Bei der Verwendung von Transaktionen in Amazon DocumentDB wird eine Transaktion innerhalb einer Sitzung mithilfe der `session.startTransaction()` API initiiert, und eine Sitzung unterstützt jeweils eine einzelne Transaktion. Ebenso werden Transaktionen entweder mit den APIs commit (`session.commitTransaction()`) oder abort (`session.abortTransaction()`) abgeschlossen.

Kausale Konsistenz

Die kausale Konsistenz garantiert, dass der Client innerhalb einer einzelnen Clientsitzung die Konsistenz von Lesen nach Schreiben einhält. Monatomisch reads/writes, und Schreibvorgänge folgen Lesevorgängen. Diese Garantien gelten für alle Instanzen in einem Cluster, nicht nur für die primäre. Amazon DocumentDB unterstützt keine kausale Konsistenz und die folgende Anweisung führt zu einem Fehler.

```
var mySession = db.getMongo().startSession();
var mySessionObject = mySession.getDatabase('test').getCollection('account');

mySessionObject.updateOne({"_id": 2}, {"$inc": {"balance": 400}});
//Result:{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

mySessionObject.find()
//Error: error: {
//         "ok" : 0,
//         "code" : 303,
//         "errmsg" : "Feature not supported: 'causal consistency'",
//         "operationTime" : Timestamp(1603461817, 493214)
//}

mySession.endSession()
```

Sie können die kausale Konsistenz innerhalb einer Sitzung deaktivieren. Beachten Sie, dass Sie auf diese Weise das Sitzungsframework nutzen können, aber keine kausale Konsistenzgarantie für Lesevorgänge bietet. Wenn Sie Amazon DocumentDB verwenden, sind Lesevorgänge von der Primärinstanz nach dem Schreiben konsistent, und die Lesevorgänge aus den Replikat-Instances sind letztendlich konsistent. Transaktionen sind der Hauptanwendungsfall für die Nutzung von Sitzungen.

```
var mySession = db.getMongo().startSession({causalConsistency: false});
var mySessionObject = mySession.getDatabase('test').getCollection('account');

mySessionObject.updateOne({"_id": 2}, {"$inc": {"balance": 400}});
//Result:{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

mySessionObject.find()
//{ "_id" : 1, "name" : "Bob", "balance" : 100 }
```

```
//{ "_id" : 2, "name" : "Alice", "balance" : 1700 }
```

Wiederholbare Schreibvorgänge

Bei wiederholbaren Schreibvorgängen wiederholt Ihr Treiber automatisch einen Schreibvorgang einmal, wenn ein Netzwerkfehler auftritt oder der Treiber den primären Schreibvorgang nicht finden kann. Ab der Engine-Version 8.0.2 unterstützt Amazon DocumentDB wiederholbare Schreibversuche. In früheren Engine-Versionen müssen wiederholbare Schreibvorgänge durch Angabe in der Verbindungszeichenfolge deaktiviert werden. `retryWrites=false` Weitere Informationen finden Sie unter Wiederholbare Schreibvorgänge in Amazon DocumentDB.

Fehler bei der Transaktion

Bei der Verwendung von Transaktionen gibt es Szenarien, in denen ein Fehler auftreten kann, der besagt, dass eine Transaktionsnummer keiner laufenden Transaktion entspricht.

Der Fehler kann in mindestens zwei verschiedenen Szenarien generiert werden:

- Nach dem Transaktions-Timeout von einer Minute.
- Nach einem Instanzneustart (aufgrund von Patches, Wiederherstellung nach einem Absturz usw.) kann es vorkommen, dass dieser Fehler auch dann auftritt, wenn die Transaktion erfolgreich ausgeführt wurde. Während eines Instanzneustarts kann die Datenbank den Unterschied zwischen einer erfolgreich abgeschlossenen und einer abgebrochenen Transaktion nicht erkennen. Mit anderen Worten, der Status des Abschlusses der Transaktion ist mehrdeutig.

Der beste Weg, diesen Fehler zu beheben, besteht darin, transaktionale Aktualisierungen idempotent zu machen — zum Beispiel, indem Sie den `$set` Mutator anstelle einer Operation verwenden. increment/decrement Siehe unten:

```
{ "ok" : 0,
"operationTime" : Timestamp(1603938167, 1),
"code" : 251,
"errmsg" : "Given transaction number 1 does not match any in-progress transactions."
}
```

Wiederholbare Schreibvorgänge in Amazon DocumentDB

Ab der Engine-Version 8.0.2 unterstützt Amazon DocumentDB wiederholbare Schreibversuche. Wenn ein Schreibvorgang aufgrund eines vorübergehenden Netzwerkfehlers oder einer Primärwahl fehlschlägt, kann der Treiber den Vorgang automatisch genau einmal wiederholen. Amazon DocumentDB dedupliziert den erneuten Schreibversuch, sodass der Vorgang höchstens einmal ausgeführt wird, wodurch die Idempotenz erhalten bleibt.

Für erneut versuchbare Schreibvorgänge ist ein MongoDB-compatible Treiber erforderlich, der das Protokoll für wiederholbare Schreibvorgänge unterstützt. Die meisten aktuellen MongoDB-compatible Treiber ermöglichen standardmäßig wiederholbare Schreibversuche (`retryWrites=true`in der Verbindungszeichenfolge). Wenn Sie ein Upgrade von einer früheren Version auf Engine 8.0.2 durchführen, können Sie dies `retryWrites=false` aus Ihrer Verbindungszeichenfolge entfernen, um diese Funktion zu aktivieren.

Voraussetzungen

Um wiederholbare Schreibvorgänge verwenden zu können, müssen Sie die folgenden Anforderungen erfüllen:

- Amazon DocumentDB-Engine Version 8.0.2 oder höher.
- Ein MongoDB-compatible Treiber, der wiederholbare Schreibversuche unterstützt. Die Mindestversion finden Sie in der Dokumentation zu Ihrem Treiber.
- Die Verbindungszeichenfolge muss Folgendes enthalten`retryWrites=true`, oder der Treiber muss standardmäßig wiederholbare Schreibvorgänge verwenden (bei den meisten aktuellen Treibern ist das der Fall).

Unterstützte Vorgänge

Wiederholbare Schreibvorgänge gelten für Schreibvorgänge, bei denen sich jeder einzelne Schreibvorgang auf höchstens ein Dokument auswirkt. Batch-Operationen kommen in Frage, da jeder Schreibvorgang innerhalb des Batches separat dedupliziert wird.

Die folgenden Schreibvorgänge können wiederholt werden:

- `insertOne`
- `insertMany`

- `updateOne`
- `deleteOne`
- `findOneAndUpdate`
- `findOneAndDelete`
- `findOneAndReplace`
- `bulkWrite`(wenn sie aus`insertOne`, `updateOne``deleteOne`, oder `replaceOne` Operationen bestehen)

Bei `insertMany` und `bulkWrite` wird jedes Dokument im Stapel einzeln dedupliziert. Wenn ein erneuter Versuch erforderlich ist, werden nur die Dokumente eingefügt, die noch nicht angewendet wurden.

> **Note**
>
> `commitTransaction`und können ebenfalls erneut versucht `abortTransaction` werden, und zwar als Befehle zur Transaktionskontrolle und nicht als Schreibvorgänge. Die einzelnen Schreibvorgänge innerhalb einer Transaktion können nicht wiederholt werden. Weitere Informationen finden Sie unter Einschränkungen.

Einschränkungen

Die folgenden Einschränkungen gelten für Schreibversuche in Amazon DocumentDB, die erneut versucht werden können:

- `updateMany`und können nicht erneut versucht `deleteMany` werden.
- Schreibvorgänge innerhalb von Transaktionen mit mehreren Anweisungen können nicht erneut versucht werden. Operationen zum Festschreiben und Abbrechen von Transaktionen können separat wiederholt werden.
- Dokumente müssen ein `_id` Feld für wiederholbare Einfügungen enthalten.
- Wiederholbare Schreibvorgänge sind nur in der Engine-Version 8.0.2 und höher verfügbar. Geben Sie `retryWrites=false` in früheren Engine-Versionen Ihre Verbindungszeichenfolge ein, um Fehler zu vermeiden.
- Schreibvorgänge können auf Clustern, die Query Planner Version 1.0 verwenden, nicht erneut versucht werden.

- Bei `findOneAndReplace` Vorgängen und Vorgängen `findOneAndUpdatefindOneAndDelete`, die große Dokumente zurückgeben, kann es bei wiederholten Schreibvorgängen zu einer erhöhten Schreiblatenz kommen, da das gesamte Ergebnisdokument zur Deduplizierung zwischengespeichert wird. Wenn die Schreibleistung entscheidend ist und die zurückgegebenen Dokumente groß sind, sollten Sie die Einstellungen für diese Workloads in Betracht ziehen. `retryWrites=false`

Wiederholbare Schreibversuche aktivieren

Um wiederholbare Schreibvorgänge zu aktivieren, fügen Sie Folgendes `retryWrites=true` in Ihre Verbindungszeichenfolge ein:

```
mongodb://<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=true
```

Wenn Ihr Treiber standardmäßig auf eingestellt ist`retryWrites=true`, können Sie alle expliziten Angaben `retryWrites=false` aus Ihrer Verbindungszeichenfolge entfernen.

Die vollständige Anleitung zur Verbindungszeichenfolge finden Sie unter. Programmgesteuertes Herstellen einer Verbindung zu Amazon DocumentDB

So funktionieren wiederholbare Schreibvorgänge

Wenn Ihre Anwendung einen Schreibvorgang mit sendet`retryWrites=true`, fügt der Treiber dem Schreibbefehl eine logische Sitzungs-ID (`lsid`) und eine Transaktionsnummer (`txnNumber`) hinzu. Amazon DocumentDB verwendet diese Identifikatoren, um Schreibvorgänge zu deduplizieren:

- Beim ersten Versuch führt Amazon DocumentDB den Schreibvorgang aus und speichert das Ergebnis im Cache.
- Wenn der Treiber denselben Schreibvorgang wiederholt (dasselbe `lsid` und`txnNumber`), gibt Amazon DocumentDB das zwischengespeicherte Ergebnis zurück, ohne den Schreibvorgang erneut auszuführen.
- Amazon DocumentDB berücksichtigt Wiederholungsversuche für 60 Minuten nach dem ursprünglichen Schreibvorgang. Die Treiber versuchen es sofort erneut, sodass dieses Fenster viel länger ist, als ein Treiber benötigt.

Wenn ein Wiederholungsversuch mehr als 60 Minuten nach dem ursprünglichen Schreibvorgang eintrifft, hat Amazon DocumentDB dafür kein zwischengespeichertes Ergebnis mehr und führt den Schreibvorgang als neuen Vorgang aus.

Fehlerbehandlung

Die folgenden Fehler sind spezifisch für wiederholbare Schreibvorgänge in Amazon DocumentDB:

Fehlercode	Name	Description
225	TransactionTooOld	Eine neuere Transaktionsnummer für dieselbe Sitzung und Aussage wurde bereits festgeschrieben, daher ist dieser Wiederholungsversuch zu alt, um ihn anzuwenden.
301	Wiederholbare Schreibvorgänge werden nicht unterstützt	Der Schreibvorgang enthielt Felder, in denen erneut versucht werden kann, aber die Engine-Version unterstützt keine wiederholbaren Schreibvorgänge, oder der Vorgang kann nicht wiederholt werden. `retryWrites=false` Wird gesetzt, wenn eine Verbindung zu Engine-Versionen vor 8.0.2 hergestellt wird.

Migration von retryWrites=False

Wenn Sie auf Engine-Version 8.0.2 aktualisieren und derzeit in Ihren Verbindungszeichenfolgen verwenden: `retryWrites=false`

1. Aktualisieren Sie Ihren Cluster auf Engine-Version 8.0.2 oder höher.
2. Entfernen Sie es `retryWrites=false` aus den Verbindungszeichenfolgen Ihrer Anwendung, oder ändern Sie es in`retryWrites=true`.
3. Es sind keine Änderungen des Anwendungscodes erforderlich. Der Treiber verarbeitet Wiederholungen automatisch.

Verwenden der Sortierung in Amazon DocumentDB 8.0

Amazon DocumentDB 8.0 unterstützt jetzt die Sortierung. Die Sortierung ermöglicht es Ihnen, sprachspezifische Regeln für den Zeichenkettenvergleich zu konfigurieren. Mit der Sortierung können Sie Regeln für Vergleiche unter Berücksichtigung der Groß- und Kleinschreibung oder ein Sprachgebietsschema angeben. Die Sortierung kann in DocumentDB 8.0 auf Sammlungs- oder Indexebene konfiguriert werden. Wenn Sie die Sortierung in DocumentDB verwenden, wird intern ein Kollatierungsdokument mit den folgenden Parametern erstellt.

```
{
locale: string,
caseLevel: boolean,
caseFirst: string,
strength: int,
numericOrdering: boolean,
alternate: string,
maxVariable: string,
backwards: boolean,
normalization: boolean
}
```

Einschränkungen

Für die Sortierung gelten in Amazon DocumentDB die folgenden Einschränkungen:

- Die Sortierung ist mit Planner v3 kompatibel, das in Amazon DocumentDB 8.0 verfügbar ist. Der Wechsel zu Plannerv2 oder Plannerv1 kann zu inkonsistentem Verhalten führen, einschließlich der Fehlermeldung „Index nicht gefunden“.
- Wenn eine Mongodb-Sammlung mit Sortierung exportiert wird, müssen Sie aufgrund inhärenter Bibliotheksunterschiede die Dateien metadata.bson aktualisieren und ihre Version vor der Migration von 57.1 auf 60.2 ändern.
- In seltenen Fällen können Ihre Sortierungseinstellungen das interne Limit für die Zeichenanzahl überschreiten und den folgenden Fehler verursachen. „Fehler: Das Sortierungsdokument enthält nicht standardmäßige Attribute, die mehr als unterstützt werden. Bitte reduzieren Sie die Anzahl der Optionen.“ Versuchen Sie in diesem Fall, die Optionen, die Sie im Kollationsdokument angeben, zu reduzieren, oder versuchen Sie, wo immer möglich, Standardwerte zu verwenden.

Verwenden von Ansichten in Amazon DocumentDB 8.0

- Amazon DocumentDB 8.0 unterstützt jetzt Ansichten. Ansichten funktionieren als virtuelle Sammlungen, die Daten basierend auf bestimmten Aggregationsvorgängen präsentieren. Wenn Sie eine Ansicht erstellen, definieren Sie eine Abfrage, die Daten aus einer oder mehreren Quellensammlungen transformiert. Amazon DocumentDB 8.0 führt diese Abfrage bei jedem Zugriff auf die Ansicht aus, ohne zusätzliche Speicherressourcen zu verbrauchen. Im Gegensatz zu Standardsammlungen speichern Ansichten in Amazon DocumentDB 8.0 keine Dokumente auf der Festplatte, was sie zu einer effizienten Lösung für die Präsentation transformierter oder gefilterter Daten für Anwendungen macht. Um eine Ansicht in Amazon DocumentDB zu erstellen, können Sie den Befehl createView oder die Hilfsmethode db.createView () verwenden:

```
db.createView("viewName","sourceCollection",
[
    { $match: { status: "active" } },
    { $project: { _id: 1, name: 1, email: 1 } }
]
)
```

  Dadurch wird eine Ansicht mit dem Namen „ViewName“ erstellt, die auf „sourceCollection“ basiert und nur aktive Dokumente und Projekte enthält, nur die Felder _id, name und email. Ansichten in Amazon DocumentDB sind schreibgeschützt. Schreibvorgänge an Ansichten geben einen Fehler zurück. Für eine optimale Leistung bei großen Datensätzen können Sie Ihre View-Pipelines so strukturieren, dass die Effizienz maximiert wird. Bei komplexen Aggregationspipelines wird empfohlen, die $match-Stufe als erste Phase oder zu Beginn der Pipeline zu verwenden, um die Anzahl der Dokumente zu reduzieren, die in den nachfolgenden Phasen verarbeitet werden müssen, und so die Abfrageleistung zu verbessern.

Bewährte Methoden

Nachfolgend sind einige bewährte Methoden aufgeführt, die Sie bei Views anwenden sollten.

- Früh filtern: Verwenden Sie $match-Stufen zu einem frühen Zeitpunkt in View-Pipelines, um die Menge der verarbeiteten Daten zu reduzieren.

- Vermeiden Sie komplexe Aggregationen: Für häufig aufgerufene Ansichten mit komplexen Aggregationen sollten Sie erwägen, eine separate Sammlung mit vorab berechneten Ergebnissen zu erstellen, die regelmäßig aktualisiert werden.
- Indexplanung: Stellen Sie sicher, dass Felder, die in View-Pipelines verwendet werden, insbesondere bei $match- und $sort-Vorgängen, in der Quellsammlung ordnungsgemäß indexiert sind.
- Abfrageoptimierung: Verwenden Sie den Befehl explain, um zu verstehen, wie Ihre View-Abfragen ausgeführt werden, und um sie entsprechend zu optimieren.
- Alternativen für Ansichten: Angesichts der funktionalen Unterschiede zwischen Amazon DocumentDB- und MongoDB-Ansichten sollten Sie bei Einschränkungen die Verwendung regulärer Sammlungen mit geplanten Aktualisierungen als Alternative zu Ansichten in Betracht ziehen.

Kompatibilität mit Aggregator-Operatoren

Amazon DocumentDB unterstützt viele Aggregationsoperatoren in Ansichtsdefinitionen und erweitert gleichzeitig die Kompatibilität weiter. Konzentrieren Sie sich bei der Verwendung von Ansichten auf die folgenden unterstützten Operatoren:

- $match zum Filtern von Dokumenten
- $project für Feldauswahl und Transformation
- $addFields zum Hinzufügen von berechneten Feldern
- $sort für die Reihenfolge der Ergebnisse
- $limit und $skip für die Paginierung

Einige spezialisierte Operatoren wie $currentOp, $replaceRoot und $geoNear funktionieren derzeit eher in direkten Aggregationsabfragen als in Ansichtsdefinitionen.

Nutzung von Indizes und Ansichten

Ansichten in Amazon DocumentDB 8.0 verwenden die Indizes der zugrunde liegenden Sammlung. Daher können Sie Indizes für eine Ansicht nicht direkt erstellen, löschen oder neu erstellen. Gut entworfene Indizes für die Quellsammlung können jedoch die Leistung von Anzeigeabfragen erheblich verbessern. Nachfolgend finden Sie einige Schritte zur Optimierung der Abfrageleistung für Ansichten:

- Stellen Sie sicher, dass für die Felder der Quellsammlung, die in der Pipeline der Ansicht verwendet werden, geeignete Indizes vorhanden sind, insbesondere bei den Vorgängen $match und $sort
- Verwenden Sie die Methode explain (), um Ausführungspläne für Abfragen zu analysieren und die Indexnutzung zu überprüfen. E.g., `db.viewName.find({...}).explain()`

Generative künstliche Intelligenz von Amazon DocumentDB

Amazon DocumentDB bietet Funktionen, mit denen Modelle für maschinelles Lernen (ML) und generativer künstlicher Intelligenz (KI) in Echtzeit mit in Amazon DocumentDB gespeicherten Daten arbeiten können. Kunden müssen keine Zeit mehr damit verbringen, eine separate Infrastruktur zu verwalten, Code für die Verbindung mit einem anderen Dienst zu schreiben und Daten aus ihrer primären Datenbank zu duplizieren.

Weitere Informationen über künstliche Intelligenz und darüber, wie Sie Ihre KI-Anforderungen unterstützen AWS können, finden Sie in diesem "What-is" Artikel.

Topics

- No-code maschinelles Lernen mit Amazon SageMaker AI Canvas
- Vektorsuche für Amazon DocumentDB

No-code maschinelles Lernen mit Amazon SageMaker AI Canvas

Amazon SageMaker AI Canvas ermöglicht es Ihnen, Ihre eigenen AI/ML Modelle zu erstellen, ohne eine einzige Codezeile schreiben zu müssen. Sie können ML-Modelle für gängige Anwendungsfälle wie Regression und Prognose erstellen und auf Fundamentmodelle (FMs) von Amazon Bedrock zugreifen und diese evaluieren. Sie können auch auf öffentliche FMs von Amazon SageMaker AI zugreifen, um Inhalte JumpStart zu generieren, Texte zu extrahieren und zusammenzufassen, um generative KI-Lösungen zu unterstützen.

Wie erstellt man ML-Modelle ohne Code mit AI Canvas SageMaker

Amazon DocumentDB ist jetzt in Amazon SageMaker AI Canvas integriert, um maschinelles Lernen (ML) ohne Code mit in Amazon DocumentDB gespeicherten Daten zu ermöglichen. Sie können jetzt ML-Modelle für Regressions- und Prognoseanforderungen erstellen und Basismodelle für die Zusammenfassung und Generierung von Inhalten verwenden, indem Sie in Amazon DocumentDB gespeicherte Daten verwenden, ohne eine einzige Codezeile schreiben zu müssen.

SageMaker AI Canvas bietet eine visuelle Oberfläche, die es Amazon DocumentDB-Kunden ermöglicht, Prognosen zu erstellen, ohne dass AI/ML Fachwissen erforderlich ist oder eine einzige Codezeile geschrieben werden muss. Kunden können den SageMaker AI Canvas-Arbeitsbereich jetzt von dort aus starten AWS-Managementkonsole, Amazon DocumentDB-Daten importieren und zusammenfügen, um Daten vorzubereiten und Modelle zu trainieren. Daten in Amazon DocumentDB können jetzt in SageMaker AI Canvas verwendet werden, um Modelle zu erstellen und zu erweitern, um Kundenabwanderung vorherzusagen, Betrug zu erkennen, Wartungsausfälle vorherzusagen, Geschäftskennzahlen zu prognostizieren und Inhalte zu generieren. Kunden können jetzt mithilfe der nativen Integration von SageMaker AI Canvas in Quick ML-driven Erkenntnisse veröffentlichen und teamübergreifend teilen. Pipelines für die Datenaufnahme in SageMaker AI Canvas werden standardmäßig auf sekundären Amazon DocumentDB-Instances ausgeführt, wodurch sichergestellt wird, dass die Leistung der Anwendungs- und SageMaker AI Canvas-Erfassungs-Workloads nicht beeinträchtigt wird.

Amazon DocumentDB-Kunden können mit SageMaker AI Canvas beginnen, indem sie zur neuen Amazon DocumentDB No-Code ML Console-Seite navigieren und eine Verbindung zu neuen oder verfügbaren AI Canvas-Arbeitsbereichen herstellen. SageMaker

Konfiguration der SageMaker AI-Domain und des Benutzerprofils

Sie können von SageMaker AI-Domänen aus, die im Modus „Nur VPC“ ausgeführt werden, eine Verbindung zu Amazon DocumentDB-Clustern herstellen. Indem Sie eine SageMaker AI-Domain in Ihrer VPC starten, können Sie den Datenfluss von Ihren SageMaker AI Studio- und Canvas-Umgebungen aus steuern. Auf diese Weise können Sie den Internetzugang einschränken, den Datenverkehr mithilfe von AWS Standardnetzwerk- und Sicherheitsfunktionen überwachen und überprüfen und über VPC-Endpunkte eine Verbindung zu anderen AWS Ressourcen herstellen. Informationen zum Erstellen Ihrer SageMaker AI-Domain für die Verbindung mit Ihrem Amazon SageMaker DocumentDB-Cluster finden Sie unter Amazon AI Canvas Erste Schritte und Konfiguration von Amazon SageMaker AI Canvas in einer VPC ohne Internetzugang im Amazon SageMaker AI Developer Guide.

Konfiguration der IAM-Zugriffsberechtigungen für Amazon DocumentDB und AI Canvas SageMaker

Ein Amazon DocumentDB-Benutzer, `AmazonDocDBConsoleFullAccess` der der ihm zugewiesenen Rolle und Identität zugeordnet ist, kann auf die zugreifen. AWS-Managementkonsole Fügen Sie der oben genannten Rolle oder Identität die folgenden Aktionen hinzu, um den Zugriff auf maschinelles Lernen ohne Code mit Amazon SageMaker AI Canvas zu ermöglichen.

```
"sagemaker:CreatePresignedDomainUrl",
"sagemaker:DescribeDomain",
"sagemaker:ListDomains",
"sagemaker:ListUserProfiles"
```

Datenbankbenutzer und Rollen für SageMaker AI Canvas erstellen

Sie können den Zugriff auf die Aktionen einschränken, die Benutzer mit Datenbanken ausführen können, indem Sie die rollenbasierte Zugriffskontrolle (RBAC) in Amazon DocumentDB verwenden. Bei RBAC werden einem Benutzer eine oder mehrere Rollen gewährt. Diese Rollen bestimmen die Operationen, die ein Benutzer mit Datenbankressourcen ausführen kann.

Als Canvas-Benutzer stellen Sie mit Anmeldeinformationen für Benutzername und Passwort eine Verbindung zu einer Amazon DocumentDB-Datenbank her. Mithilfe der RBAC-Funktionalität von Amazon DocumentDB können Sie eine Datenbank user/role für einen Canvas-Benutzer erstellen, der Lesezugriff auf die spezifischen Datenbanken hat.

Verwenden Sie zum Beispiel die folgende Operation: `createUser`

```
db.createUser({
user: "canvas_user",
pwd: "<insert-password>",
roles: [{role: "read", db: "sample-database-1"}]
})
```

Dadurch wird eine erstellt`canvas_user`, die Leseberechtigungen für die `sample-database-1` Datenbank hat. Ihre Canvas-Analysten können diese Anmeldeinformationen verwenden, um auf Daten in Ihrem Amazon DocumentDB-Cluster zuzugreifen. Weitere Informationen finden Datenbankzugriff mit Role-Based Access Control Sie unter.

Verfügbare Regionen

Die No-Code-Integration ist in Regionen verfügbar, in denen sowohl Amazon DocumentDB als auch Amazon SageMaker AI Canvas unterstützt werden. Zu den Regionen gehören:

- us-east-1 (N. Virginia)
- us-east-2 (Ohio)
- us-west-2 (Oregon)
- ap-northeast-1 (Tokyo)

- ap-northeast-2 (Seoul)
- ap-south-1 (Mumbai)
- ap-southeast-1 (Singapur)
- ap-southeast-2 (Sydney)
- eu-central-1 (Frankfurt)
- eu-west-1 (Ireland)

Die aktuelle Verfügbarkeit der Regionen finden Sie im Amazon SageMaker AI Developer Guide unter Amazon AI Canvas. SageMaker

Vektorsuche für Amazon DocumentDB

Die Vektorsuche ist eine Methode, die beim maschinellen Lernen verwendet wird, um ähnliche Datenpunkte zu einem bestimmten Datenpunkt zu finden, indem ihre Vektordarstellungen mithilfe von Entfernungs- oder Ähnlichkeitsmetriken verglichen werden. Je näher sich die beiden Vektoren im Vektorraum befinden, desto ähnlicher werden die zugrunde liegenden Elemente betrachtet. Diese Technik hilft dabei, die semantische Bedeutung der Daten zu erfassen. Dieser Ansatz ist in verschiedenen Anwendungen nützlich, z. B. in Empfehlungssystemen, bei der Verarbeitung natürlicher Sprache und bei der Bilderkennung.

Die Vektorsuche für Amazon DocumentDB kombiniert die Flexibilität und die umfassenden Abfragefunktionen einer JSON-based Dokumentendatenbank mit der Leistungsfähigkeit der Vektorsuche. Wenn Sie Ihre vorhandenen Amazon DocumentDB-Daten oder eine flexible Dokumentdatenstruktur verwenden möchten, um Anwendungsfälle für maschinelles Lernen und generative KI zu erstellen, wie z. B. semantische Suchfunktionen, Produktempfehlungen, Personalisierung, Chatbots, Betrugserkennung und Anomalieerkennung, dann ist die Vektorsuche für Amazon DocumentDB die ideale Wahl für Sie. Die Vektorsuche ist in instanzbasierten Amazon DocumentDB 5.0-Clustern verfügbar.

Topics

- Vektoren einfügen
- Erstellen eines Vektorindexes
- Eine Indexdefinition abrufen
- Abfragen von Vektoren
- Funktionen und Einschränkungen

- Bewährte Methoden

Vektoren einfügen

Um Vektoren in Ihre Amazon DocumentDB-Datenbank einzufügen, können Sie die vorhandenen Einfügemethoden verwenden:

Beispiel

Im folgenden Beispiel wird eine Sammlung von fünf Dokumenten in einer Testdatenbank erstellt. Jedes Dokument enthält zwei Felder: den Produktnamen und die entsprechende Vektoreinbettung.

```
db.collection.insertMany([
  {"product_name": "Product A", "vectorEmbedding": [0.2, 0.5, 0.8]},
  {"product_name": "Product B", "vectorEmbedding": [0.7, 0.3, 0.9]},
  {"product_name": "Product C", "vectorEmbedding": [0.1, 0.2, 0.5]},
  {"product_name": "Product D", "vectorEmbedding": [0.9, 0.6, 0.4]},
  {"product_name": "Product E", "vectorEmbedding": [0.4, 0.7, 0.2]}
]);
```

Erstellen eines Vektorindexes

Amazon DocumentDB unterstützt sowohl die Indexierung von Hierarchical Navigable Small World (HNSW) als auch die Indexierungsmethoden Inverted File with Flat Compression (IVFflat). Ein IVFFlat-Index unterteilt Vektoren in Listen und durchsucht anschließend eine ausgewählte Teilmenge der Listen, die dem Abfragevektor am nächsten sind. Andererseits organisiert ein HNSW-Index die Vektordaten in einem vielschichtigen Diagramm. HNSW hat zwar im Vergleich zu IVFflat langsamere Erstellungszeiten, bietet aber eine bessere Abfrageleistung und einen besseren Abruf. Im Gegensatz zu IVFflat ist bei HNSW kein Trainingsschritt erforderlich, sodass der Index ohne anfängliches Laden der Daten generiert werden kann. Verwenden Sie für die meisten Anwendungsfälle den HNSW-Indextyp für die Vektorsuche.

Wenn Sie keinen Vektorindex erstellen, führt Amazon DocumentDB eine exakte Suche nach dem nächsten Nachbarn durch und gewährleistet so einen perfekten Abruf. In Produktionsszenarien ist Geschwindigkeit jedoch entscheidend. Verwenden Sie Vektorindizes, die einen gewissen Rückruf gegen eine höhere Geschwindigkeit eintauschen könnten. Es ist wichtig zu beachten, dass das Hinzufügen eines Vektorindexes zu unterschiedlichen Abfrageergebnissen führen kann.

Vorlagen

Sie können Folgendes `createIndex` oder `runCommand` Vorlagen verwenden, um einen Vektorindex für ein Vektorfeld zu erstellen:

Using createIndex

In bestimmten Treibern wie Mongosh und Java `createIndex` kann die Verwendung der `vectorOptions` Parameter in zu einem Fehler führen. Verwenden `runCommand` Sie in solchen Fällen:

```
db.collection.createIndex(
  { "<vectorField>": "vector" },
  { "name": "<indexName>",
    "vectorOptions": {
      "type": " <hnsw> | <ivfflat> ",
      "dimensions": <number_of_dimensions>,
      "similarity": " <euclidean> | <cosine> | <dotProduct> ",
      "lists": <number_of_lists> [applicable for IVFFlat],
      "m": <max number of connections> [applicable for HNSW],
      "efConstruction": <size of the dynamic list for index build> [applicable for
 HNSW]
    }
  }
);
```

Using runCommand

In bestimmten Treibern wie Mongosh und Java `createIndex` kann die Verwendung der `vectorOptions` Parameter in zu einem Fehler führen. Verwenden `runCommand` Sie in solchen Fällen:

```
db.runCommand(
  { "createIndexes": "<collection>",
  "indexes": [{
      key: { "<vectorField>": "vector" },
      vectorOptions: {
          type: " <hnsw> | <ivfflat> ",
          dimensions: <number of dimensions>,
          similarity: " <euclidean> | <cosine> | <dotProduct> ",
          lists: <number_of_lists> [applicable for IVFFlat],
          m: <max number of connections> [applicable for HNSW],
          efConstruction: <size of the dynamic list for index build> [applicable for
 HNSW]
```

```
            },
        name: "myIndex"
        }]
    }
);
```

Parameter	Anforderung	Datentyp	Description	Wert (e)
name	optional	Zeichenfolge	Gibt den Namen des Indexes an.	Alphanumerisch
type	optional		Gibt den Indextyp an.	Unterstützt: hnsw oder ivfflat Standard: HNSW (Engine-Patch 3.0.4574 ab)
dimensions	Erforderlich	Ganzzahl	Gibt die Anzahl der Dimensionen in den Vektordaten an.	Maximal 2.000 Dimensionen.
similarity	Erforderlich	Zeichenfolge	Gibt die Entfernungsmetrik an, die für die Ähnlichkeitsberechnung verwendet wird.	• **euclidean** • **cosine** • **dotProduct**
lists	erforderlich für IVFflat	Ganzzahl	Gibt die Anzahl der Cluster an, die der IVFFlat-Index verwendet, um die Vektordaten zu gruppieren. Die empfohlene Einstellung ist	Minimum: 1 Maximum: Weitere Informationen finden Sie in der Tabelle für die einzelnen Instanztypen

Parameter	Anforderung	Datentyp	Description	Wert (e)
			# von documents /1000 für bis zu 1 Million Dokumente und `sqrt(# of documents)` für über 1 Million Dokumente.	Funktionen und Einschränkungen unten.
`m`	optional	Ganzzahl	Gibt die maximale Anzahl von Verbindungen für einen HNSW-Index an	Standard: 16 Bereich [2, 100]
`efConstruction`	optional	Ganzzahl	Gibt die Größe der dynamischen Kandidatenliste für die Erstellung des Diagramms für den HNSW-Index an. `efConstruction` muss größer oder gleich (2 * m) sein	Standard: 64 Bereich [4, 1000]

Es ist wichtig, dass Sie den Wert von Unterparametern wie `lists` für ivfFlat `m` und `efConstruction` für HNSW entsprechend festlegen accuracy/recall, da sich dies auf die Erstellungszeit und die Leistung Ihrer Suche auswirkt. Ein höherer Listenwert erhöht die Geschwindigkeit der Abfrage, da dadurch die Anzahl der Vektoren in jeder Liste reduziert wird, was zu kleineren Regionen führt. Eine kleinere Regionsgröße kann jedoch zu mehr Abruffehlern führen, was zu einer geringeren Genauigkeit führt. Bei HNSW erhöht eine Erhöhung des Werts von `m` und

die Genauigkeit, `efConstruction` erhöht aber auch die Zeit und Größe der Indexerstellung. Im Folgenden sind einige Beispiele aufgeführt:

Beispiele

HNSW

```
db.collection.createIndex(
  { "vectorEmbedding": "vector" },
  { "name": "myIndex",
    "vectorOptions": {
      "type": "hnsw",
      "dimensions": 3,
      "similarity": "euclidean",
      "m": 16,
      "efConstruction": 64
    }
  }
);
```

IVFFlat

```
db.collection.createIndex(
  { "vectorEmbedding": "vector" },
  { "name": "myIndex",
    "vectorOptions": {
      "type": "ivfflat",
      "dimensions": 3,
      "similarity": "euclidean",
      "lists":1
    }
  }
)
```

Eine Indexdefinition abrufen

Sie können die Details Ihrer Indizes, einschließlich Vektorindizes, mit dem `getIndexes` folgenden Befehl anzeigen:

Beispiel

```
db.collection.getIndexes()
```

Beispielausgabe

```
[
 {
  "v" : 4,
  "key" : {
   "_id" : 1
  },
  "name" : "_id_",
  "ns" : "test.collection"
 },
 {
  "v" : 4,
  "key" : {
   "vectorEmbedding" : "vector"
  },
  "name" : "myIndex",
  "vectorOptions" : {
   "type" : "ivfflat",
   "dimensions" : 3,
   "similarity" : "euclidean",
   "lists" : 1
  },
  "ns" : "test.collection"
 }
]
```

Abfragen von Vektoren

Amazon DocumentDB unterstützt zwei Vektor-Suchoperatoren für die Abfrage von Vektoren:

Klassischer Vektor-Suchoperator

Verwenden Sie die folgende Vorlage, um einen Vektor abzufragen:

```
db.collection.aggregate([
  {
    $search: {
      "vectorSearch": {
        "vector": <query vector>,
        "path": "<vectorField>",
```

```
        "similarity": "<distance metric>",
        "k": <number of results>,
        "probes":<number of probes> [applicable for IVFFlat],
        "efSearch":<size of the dynamic list during search> [applicable for HNSW]
      }
    }
  }
]);
```

Parameter	Anforderung	Typ	Description	Wert (e)
vectorSearch	Erforderlich	operator	Wird im $search-Befehl verwendet, um die Vektoren abzufragen.	
vector	Erforderlich	Array	Gibt den Abfragevektor an, der verwendet wird, um ähnliche Vektoren zu finden.	
path	Erforderlich	Zeichenfolge	Definiert den Namen des Vektorfeldes.	
k	Erforderlich	Ganzzahl	Gibt die Anzahl der Ergebnisse an, die die Suche zurückgibt.	
similarity	Erforderlich	Zeichenfolge	Gibt die Entfernungsmetrik an, die für die Ähnlichke	• **euclidean** • **cosine** • **dotProduct**

Parameter	Anforderung	Typ	Description	Wert (e)
			itsberechnung verwendet wird.	
`probes`	optional	Ganzzahl	Die Anzahl der Cluster, die von der Vektorsuche untersucht werden sollen. Ein höherer Wert ermöglicht einen besseren Abruf auf Kosten der Geschwindigkeit. Er kann auf die Anzahl der Listen für die genaue Suche nach dem nächsten Nachbarn gesetzt werden (ab diesem Zeitpunkt verwendet der Planer den Index nicht mehr). Die empfohlene Einstellung, um mit der Feinabstimmung zu beginnen, ist`sqrt(# of lists)`.	Standard: 1

Parameter	Anforderung	Typ	Description	Wert (e)
`efSearch`	optional	Ganzzahl	Gibt die Größe der dynamisch en Kandidate nliste an, die der HNSW-Index bei der Suche verwendet. Ein höherer Wert von `efSearch` ermöglicht einen besseren Abruf auf Kosten der Geschwindigkeit.	Standard: 40 Bereich [1, 1000]

Es ist wichtig, den Wert von `efSearch` (HNSW) oder `probes` (IVFflat) genau einzustellen, um die gewünschte Leistung und Genauigkeit zu erzielen. Sehen Sie sich die folgenden Beispieloperationen an:

HNSW

```
db.collection.aggregate([
  {
    $search: {
      "vectorSearch": {
        "vector": [0.2, 0.5, 0.8],
        "path": "vectorEmbedding",
        "similarity": "euclidean",
        "k": 2,
        "efSearch": 40
      }
    }
  }
]);
```

IVFFlat

```
db.collection.aggregate([
```

```
  {
    $search: {
      "vectorSearch": {
        "vector": [0.2, 0.5, 0.8],
        "path": "vectorEmbedding",
        "similarity": "euclidean",
        "k": 2,
        "probes": 1
      }
    }
  }
]);
```

Beispielausgabe

Die Ausgabe dieser Operation sieht in etwa wie folgt aus:

```
{ "_id" : ObjectId("653d835ff96bee02cad7323c"), "product_name" : "Product A",
 "vectorEmbedding" : [ 0.2, 0.5, 0.8 ] }
{ "_id" : ObjectId("653d835ff96bee02cad7323e"), "product_name" : "Product C",
 "vectorEmbedding" : [ 0.1, 0.2, 0.5 ] }
```

$VectorSearch-Operator (ab Amazon DocumentDB 8.0 verfügbar)

Verwenden Sie die folgende Vorlage, um einen Vektor abzufragen:

```
db.collection.aggregate([
{
  "$vectorSearch": {
    "exact": true | false,
    "index": "<index-name>" [supports only HNSW index],
    "limit": <number-of-results> [same as k],
    "path": "<vector field-to-search>",
    "queryVector": <array-of-numbers>,
    "numCandidates": <number-of-candidates> [same as efSearch],
  }
}])
```

Funktionen und Einschränkungen

Versionskompatibilität

- Die Vektorsuche für Amazon DocumentDB ist nur in instanzbasierten Clustern von Amazon DocumentDB 5.0+ verfügbar.

Vektoren

- Amazon DocumentDB kann Vektoren mit bis zu 2.000 Dimensionen indizieren. Bis zu 16.000 Dimensionen können jedoch ohne Index gespeichert werden.

Indizes

- Für die Erstellung eines IVFFlat-Indexes ist die empfohlene Einstellung für den Parameter Listen die Anzahl von documents/1000 für bis zu 1 Million Dokumente und `sqrt(# of documents)` für über 1 Million Dokumente. Aufgrund einer Arbeitsspeicherbeschränkung unterstützt Amazon DocumentDB je nach Anzahl der Dimensionen einen bestimmten Maximalwert des Listenparameters. Zu Ihrer Information enthält die folgende Tabelle die Maximalwerte der Listenparameter für Vektoren mit 500, 1000 und 2.000 Dimensionen:

Instance-Typ	Listen mit 500 Dimensionen	Listen mit 1000 Dimensionen	Listen mit 2000 Dimensionen
t3.med	372	257	150
r5l	915	741	511
r 5, xl	1.393	1.196	901
r5,2xl	5.460	5.230	4.788
r5,4xl	7.842	7.599	7.138
r 5,8 xl	11.220	10.974	10.498
r5,12xl	13.774	13.526	13.044
r5,16xl	15.943	15.694	15.208
r5,24xl	19.585	19.335	18.845

- Bei Vektorindizes `partial` werden keine anderen Indexoptionen wie`compound`, `sparse` oder unterstützt.
- Die parallele Indexerstellung wird für den HNSW-Index in Amazon DocumentDB 5.0 nicht unterstützt.

Vektor-Abfrage

- Bei einer Vektor-Suchanfrage ist es wichtig, die Parameter genau abzustimmen, z. `probes` B. `efSearch` für optimale Ergebnisse. Je höher der Wert von `probes` oder der `efSearch` Parameter, desto höher der Abruf und desto niedriger die Geschwindigkeit. Die empfohlene Einstellung, um mit der Feinabstimmung der Sondenparameter zu beginnen, ist`sqrt(# of lists)`.

Bewährte Methoden

Erfahren Sie mehr über bewährte Methoden für die Arbeit mit der Vektorsuche in Amazon DocumentDB. Dieser Abschnitt wird fortlaufend aktualisiert, wenn neue bewährte Methoden identifiziert werden.

- Die Indexerstellung von Inverted File with Flat Compression (IVFflat) beinhaltet das Clustern und Organisieren der Datenpunkte auf der Grundlage von Ähnlichkeiten. Damit ein Index effektiver ist, laden Sie einige Daten, bevor Sie den Index erstellen.
- Bei Vektor-Suchanfragen ist es wichtig, die Parameter wie „`probes`oder“ genau abzustimmen, um `efSearch` optimale Ergebnisse zu erzielen. Je höher der Wert des `efSearch` Parameters `probes` oder, desto höher ist der Abruf und desto niedriger ist die Geschwindigkeit. Die empfohlene Einstellung, um mit der Feinabstimmung des `probes` Parameters zu beginnen, ist`sqrt(lists)`.

Ressourcen

- Die Vektorsuche für Amazon DocumentDB ist jetzt allgemein im AWS Blog verfügbar
- Beispiel für einen semantischen Suchcode
- Beispiele für Amazon DocumentDB-Vektor-Suchcodes

Verwenden AWS Lambda mit Change-Streams

Amazon DocumentDB ist integriert, AWS Lambda sodass Sie Lambda-Funktionen verwenden können, um Datensätze in einem Change Stream zu verarbeiten. Die Lambda-Ereignisquellenzuordnung ist eine Ressource, die zum Aufrufen von Lambda-Funktionen zur Verarbeitung von Amazon DocumentDB-Ereignissen verwendet werden kann, die Lambda nicht direkt aufrufen. Mit dem Amazon DocumentDB-Change-Stream als Ereignisquelle können Sie ereignisgesteuerte Anwendungen erstellen, die auf Änderungen in Ihren Daten reagieren. Sie können beispielsweise Lambda-Funktionen verwenden, um neue Dokumente zu verarbeiten, Aktualisierungen vorhandener Dokumente nachzuverfolgen oder gelöschte Dokumente zu protokollieren.

Sie können eine Ereignisquellenzuordnung konfigurieren, um Datensätze aus Ihrem Amazon DocumentDB-Change-Stream an eine Lambda-Funktion zu senden. Ereignisse können einzeln oder zur Verbesserung der Effizienz gebündelt gesendet werden und werden der Reihe nach verarbeitet. Sie können das Stapelverhalten Ihrer Ereignisquellenzuordnung auf der Grundlage einer bestimmten Zeitfensterdauer (0-300 Sekunden) oder der Anzahl der Batch-Datensätze (Höchstgrenze von 10.000 Datensätzen) konfigurieren. Sie können mehrere Eventquellen-Mappings erstellen, um dieselben Daten mit mehreren Lambda-Funktionen zu verarbeiten, oder um unterschiedliche Elemente aus mehreren Streams mit einer einzigen Funktion zu verarbeiten.

Wenn Ihre Funktion einen Fehler zurückgibt, versucht Lambda den Batch erneut, bis er erfolgreich verarbeitet wird. Falls die Ereignisse im Change Stream abgelaufen sind, deaktiviert Lambda die Zuordnung der Ereignisquelle. In diesem Fall können Sie ein neues Eventquellen-Mapping erstellen und es mit einer Startposition Ihrer Wahl konfigurieren. Lambda-Ereignisquellenzuordnungen verarbeiten Ereignisse aufgrund der verteilten Natur ihrer Poller mindestens einmal. Infolgedessen kann Ihre Lambda-Funktion in seltenen Situationen doppelte Ereignisse erhalten. Folgen Sie den bewährten Methoden für die Arbeit mit AWS Lambda Funktionen und erstellen Sie idempotente Funktionen, um Probleme im Zusammenhang mit doppelten Ereignissen zu vermeiden. Weitere Informationen finden Sie im Entwicklerhandbuch unter Verwenden AWS Lambda console mit Amazon DocumentDB. AWS Lambda

Gemäß bewährten Methoden für die Leistung muss die Lambda-Funktion kurzlebig sein. Um unnötige Verarbeitungsverzögerungen zu vermeiden, sollte sie auch keine komplexe Logik ausführen. Insbesondere bei einem Hochgeschwindigkeits-Stream ist es besser, asynchrone Nachbearbeitungs-Schrittfunktions-Workflows auszulösen als synchrone Lambdas mit langer Laufzeit. Weitere Informationen AWS Lambda dazu finden Sie im AWS Lambda Entwicklerhandbuch.

Einschränkungen

Die folgenden Einschränkungen sollten Sie bei der Arbeit mit Amazon DocumentDB berücksichtigen und AWS Lambda:

- AWS Lambda wird derzeit nur auf Amazon DocumentDB 4.0 und 5.0 unterstützt.
- AWS Lambda wird derzeit nicht auf elastischen Clustern oder globalen Clustern unterstützt.
- AWS Lambda Die Nutzlastgröße darf 6 MB nicht überschreiten. Weitere Informationen zu Lambda-Batchgrößen finden Sie im Developer Guide im Abschnitt „Zuordnungen von Lambda-Ereignisquellen“ unter „Batchverhalten“. AWS Lambda

Amazon DocumentDB-Kompatibilität mit MongoDB

Amazon DocumentDB unterstützt MongoDB-Kompatibilität, einschließlich MongoDB 4.0, MongoDB 5.0 und MongoDB 8.0. MongoDB-Kompatibilität bedeutet, dass ein Großteil der Anwendungen, Treiber und Tools, die Sie heute bereits mit Ihren MongoDB-Datenbanken verwenden, mit nur geringen oder gar keinen Änderungen mit Amazon DocumentDB verwendet werden können. In diesem Abschnitt wird alles beschrieben, was Sie über die Kompatibilität von Amazon DocumentDB mit MongoDB wissen müssen, einschließlich neuer Funktionen und Funktionen, der ersten Schritte, der Migrationspfade und der Funktionsunterschiede.

Themen

- MongoDB 8.0-Kompatibilität
- MongoDB 5.0-Kompatibilität
- MongoDB 4.0-Kompatibilität
- Funktionale Unterschiede: Amazon DocumentDB und MongoDB
- Unterstützte MongoDB-APIs, Operationen und Datentypen in Amazon DocumentDB

MongoDB 8.0-Kompatibilität

Themen

- Was ist neu in Amazon DocumentDB 8.0
- Beginnen Sie mit Amazon DocumentDB 8.0
- Führen Sie ein Upgrade durch oder migrieren Sie zu Amazon DocumentDB 5.0 oder 8.0
- Funktionelle Unterschiede

Was ist neu in Amazon DocumentDB 8.0

Amazon DocumentDB 8.0 verbessert die Abfrageleistung um das bis zu 7-fache und das Komprimierungsverhältnis um das bis zu 5-fache, sodass Sie Hochleistungsanwendungen zu geringeren Kosten erstellen können. Amazon DocumentDB 8.0 ist vollständig mit MongoDB 8.0 kompatibel. In der folgenden Zusammenfassung werden einige der wichtigsten Funktionen vorgestellt, die in Amazon DocumentDB 8.0 eingeführt wurden. Eine vollständige Liste der neuen Funktionen finden Sie imVersionshinweise.

- Bietet Kompatibilität mit MongoDB 8.0 durch Hinzufügen von Unterstützung für MongoDB 8.0-API-Treiber. Unterstützt auch Anwendungen, die mit den MongoDB-API-Versionen 6.0 und 7.0 erstellt wurden.
- Der neue Abfrageplaner in Amazon DocumentDB 8.0 erweitert die Leistungsverbesserungen auf Operatoren der Aggregationsphase und unterstützt außerdem Optimierungen der Aggregationspipeline und verschiedene Befehle.
- Amazon DocumentDB 8.0 unterstützt Sortierung und Ansichten.
- Bietet 6 neue Aggregationsstufen: $replaceWith, $VectorSearch, $merge, $set, $unset, $bucket und 3 neue Aggregationsoperatoren $pow, $rand, $dateTrunc.
- Eine neue Version des Textindex: Der Textindex v2 in Amazon DocumentDB 8.0 führt zusätzliche Token ein und erweitert so die Textsuchfunktionen.
- Durch die parallele Erstellung von Vektorindizes reduziert Amazon DocumentDB 8.0 die Zeit für die Indexerstellung um das bis zu 30-fache.
- Das Speicherlimit wurde auf 256 TiB erhöht.

Beginnen Sie mit Amazon DocumentDB 8.0

Informationen zu den ersten Schritten mit Amazon DocumentDB 8.0 finden Sie in der. Erste Schritte mit Amazon DocumentDB Sie können einen neuen Amazon DocumentDB 8.0-Cluster mit dem AWS-Managementkonsole oder dem AWS SDK AWS CLI, oder erstellen. CloudFormation Wenn Sie eine Verbindung zu Amazon DocumentDB herstellen, müssen Sie einen MongoDB-Treiber oder ein Hilfsprogramm verwenden, das mit MongoDB 5.0 oder höher kompatibel ist.

> **Note**
>
> Wenn Sie das AWS SDK oder CloudFormation verwenden AWS CLI, wird die Engine-Version standardmäßig auf 5.0.0 gesetzt. Sie müssen den Parameter explizit angeben, `engineVersion = 8.0.0` um einen neuen Amazon DocumentDB 8.0-Cluster oder `engineVersion = 4.0.0` einen neuen Amazon DocumentDB 4.0-Cluster oder einen neuen Amazon DocumentDB 3.6-Cluster `engineVersion = 3.6.0` zu erstellen. Für einen bestimmten Amazon DocumentDB-Cluster können Sie die Cluster-Version mithilfe von AWS CLI to call ermitteln `describe-db-clusters` oder die Amazon DocumentDB-Managementkonsole verwenden, um die Engine-Versionsnummer für einen bestimmten Cluster anzuzeigen.

Amazon DocumentDB 5.0 und 8.0 unterstützen den Amazon EC2 `r8g` Graviton4-Prozessor und Graviton2-Prozessoren wie und `t4.medium` Instance-Typen für Ihre Cluster `r6g` und sind in allen unterstützten Versionen verfügbar (siehe). AWS-Regionen Instances Weitere Informationen zur Preisgestaltung finden Sie unter Preise für Amazon DocumentDB (mit MongoDB-Kompatibilität).

Führen Sie ein Upgrade durch oder migrieren Sie zu Amazon DocumentDB 5.0 oder 8.0

Sie können von MongoDB 3.6 oder MongoDB 4.0 zu Amazon DocumentDB 5.0 oder Amazon DocumentDB 8.0 migrieren, indem Sie Hilfsprogramme wie mongodump, mongorestore, mongoimport und mongoexport verwenden. AWS DMS Anweisungen Aktualisieren Ihres Amazon DocumentDB-Clusters mit AWS Database Migration Service zur Migration finden Sie unter.

Funktionelle Unterschiede

Funktionale Unterschiede zwischen Amazon DocumentDB 5.0 und 8.0

Mit der Veröffentlichung von Amazon DocumentDB 8.0 gibt es funktionale Unterschiede zwischen Amazon DocumentDB 5.0 und Amazon DocumentDB 8.0:

- Planner v1 ist der Standard-Abfrageplaner in Amazon DocumentDB 5.0, wohingegen der leistungsstärkere Planner v3 die Standardeinstellung in Amazon DocumentDB 8.0 ist.
- Neue Funktionen in Amazon DocumentDB 8.0, darunter Views, Collation und Operatoren wie $merge, sind nur mit Planner v3 kompatibel.
- Die Komprimierung ist in Amazon DocumentDB 8.0 standardmäßig aktiviert und auf die Verwendung des Zstandard-Algorithmus eingestellt. Außerdem ist „aktiviert“ in Amazon DocumentDB 8.0 keine gültige Wahl mehr. Sie können zwischen Zstd, LZ4 und None wählen.
- Amazon DocumentDB 8.0 unterstützt nur TLS 1.2 und höher für Datenbankverbindungen.

> **Important**
>
> Amazon DocumentDB 8.0 benötigt TLS 1.2 oder höher. TLS 1.0 und TLS 1.1 werden nicht mehr unterstützt.

MongoDB 5.0-Kompatibilität

Themen

- Was ist neu in Amazon DocumentDB 5.0
- Was ist neu in Amazon DocumentDB 5.0.1
- Beginnen Sie mit Amazon DocumentDB 5.0
- Führen Sie ein Upgrade durch oder migrieren Sie zu Amazon DocumentDB 5.0
- Funktionelle Unterschiede

Was ist neu in Amazon DocumentDB 5.0

Amazon DocumentDB 5.0 führt neue Funktionen und Fähigkeiten ein, zu denen Speicherbeschränkungen und clientseitige Verschlüsselung auf Feldebene gehören. In der folgenden Zusammenfassung werden einige der wichtigsten Funktionen vorgestellt, die in Amazon DocumentDB 5.0 eingeführt wurden. Eine vollständige Liste der neuen Funktionen finden Sie imVersionshinweise.

- Das Speicherlimit wurde für alle instanzbasierten Amazon DocumentDB-Cluster und shard-basierten elastischen Cluster auf 128 TiB erhöht.
- Amazon DocumentDB 5.0 Engine (Version 3.0.775) eingeführt
  - Unterstützung für MongoDB 5.0-API-Treiber
  - Unterstützung für Verschlüsselung auf Client-side Feldebene (FLE). Sie können jetzt Felder auf der Clientseite verschlüsseln, bevor Sie die Daten in den Amazon DocumentDB-Cluster schreiben. Weitere Informationen finden Sie unter Client-side Verschlüsselung auf Feldebene.
  - Neue Aggregationsoperatoren:, `$dateAdd` `$dateSubtract`
  - Unterstützt Indizes mit `$elemMatch` Operator. Daher führen Abfragen, die ausgeführt wurden`$elemMatch`, zu Index-Scans.

Amazon DocumentDB unterstützt nicht alle MongoDB 5.0-Funktionen. Bei der Entwicklung von Amazon DocumentDB 5.0 haben wir von den Funktionen und Funktionen, um deren Entwicklung uns unsere Kunden am häufigsten gebeten haben, rückwärts gearbeitet. Wir werden weiterhin zusätzliche MongoDB 5.0-Funktionen hinzufügen, je nachdem, was unsere Kunden von uns erwarten. Die aktuelle Liste der unterstützten APIs finden Sie unterUnterstützte MongoDB-APIs, Operationen und Datentypen in Amazon DocumentDB.

Was ist neu in Amazon DocumentDB 5.0.1

Amazon DocumentDB 5.0.1 ist die erste Nebenversion für die Hauptversion von Amazon DocumentDB 5.0. Nebenversionen enthalten Verbesserungen wie neue Funktionen, Sicherheitsupdates und Bugfixes. Sie können mit einem Nebenversions-Upgrade von 5.0.0 auf 5.0.1 aktualisieren. Weitere Informationen finden Sie unter Upgrade der Nebenversion von Amazon DocumentDB.

- Verbesserte Aggregationsfunktionen mit neuen Operatoren (`$rand`,,`$pow`,`$dateToParts`). `$dateFromParts`
- Metrik für aktive Verbindungen zur Überwachung von Instanzen. Weitere Informationen finden Sie unter Kontingente für Instanzen.
- Granulare Leistungsmetriken auf Befehlsebene CloudWatch für CRUD-Operationen (`find`,,,,`insert`,`findAndModify`,`update`,,`delete`,) `aggregatecount`,,`distinct`,`getMore`,`abortTransaction`,`commitTransaction`) `AvgDurationP100Duration`, Offenlegung und `MaxConcurrent` Dimensionen für eine verbesserte Betriebsüberwachung.
- Amazon DocumentDB 5.0.1 unterstützt nur TLS 1.2 und höher für Datenbankverbindungen.
- Metriken auf Sammlungsebene wurden hinzugefügt `gcRuntimeStats` und nach `storageSegmentBase` und `storageSegmentExtended` nach aufgeteilt, `collstats` um mehr Informationen über den Gesamtzustand des Garbage Collection-Prozesses zu erhalten. `documentFragmentStats`

Beginnen Sie mit Amazon DocumentDB 5.0

Informationen zu den ersten Schritten mit Amazon DocumentDB 5.0 finden Sie im. Erste Schritte mit Amazon DocumentDB Sie können einen neuen Amazon DocumentDB 5.0-Cluster mit dem AWS-Managementkonsole oder dem AWS SDK AWS CLI, oder erstellen. CloudFormation Wenn Sie eine Verbindung zu Amazon DocumentDB herstellen, müssen Sie einen MongoDB-Treiber oder ein Hilfsprogramm verwenden, das mit MongoDB 5.0 oder höher kompatibel ist.

> **Note**
>
> Wenn Sie das AWS SDK oder CloudFormation verwenden AWS CLI, wird die Engine-Version standardmäßig auf 5.0.0 gesetzt. Sie müssen den Parameter explizit angeben`engineVersion = 4.0.0`, um einen neuen Amazon DocumentDB 4.0-Cluster

oder einen neuen Amazon DocumentDB 3.6-Cluster `engineVersion = 3.6.0` zu erstellen. Für einen bestimmten Amazon DocumentDB-Cluster können Sie die Cluster-Version ermitteln, indem Sie die Amazon DocumentDB-Managementkonsole aufrufen `describe-db-clusters` oder die Amazon DocumentDB-Managementkonsole verwenden, um die Engine-Versionsnummer für einen bestimmten Cluster anzuzeigen. AWS CLI

Amazon DocumentDB 5.0 und 8.0 unterstützen den Amazon EC2 `r8g` Graviton4-Prozessor und Graviton2-Prozessoren wie und `t4.medium` Instance-Typen für Ihre Cluster `r6g` und sind in allen unterstützten Versionen verfügbar (siehe). AWS-Regionen Instances Weitere Informationen zur Preisgestaltung finden Sie unter Preise für Amazon DocumentDB (mit MongoDB-Kompatibilität).

Führen Sie ein Upgrade durch oder migrieren Sie zu Amazon DocumentDB 5.0

Sie können von MongoDB 3.6 oder MongoDB 4.0 zu Amazon DocumentDB 5.0 migrieren, indem Sie Dienstprogramme wie mongodump, mongorestore, mongoimport und mongoexport verwenden. AWS DMS Anweisungen Aktualisieren Ihres Amazon DocumentDB-Clusters mit AWS Database Migration Service zur Migration finden Sie unter.

Funktionelle Unterschiede

Funktionale Unterschiede zwischen Amazon DocumentDB 4.0 und 5.0

Mit der Veröffentlichung von Amazon DocumentDB 5.0 gibt es funktionale Unterschiede zwischen Amazon DocumentDB 4.0 und Amazon DocumentDB 5.0:

- Die integrierte Backup-Rolle unterstützt jetzt. `serverStatus` Aktion — Entwickler und Anwendungen mit Backup-Rolle können Statistiken über den Status des Amazon DocumentDB-Clusters sammeln.
- Das `SecondaryDelaySecs` Feld wird `slaveDelay` in der `replSetGetConfig` Ausgabe ersetzt.
- Der hello Befehl replace `isMaster` — hello gibt ein Dokument zurück, das die Rolle eines Amazon DocumentDB-Clusters beschreibt.
- Amazon DocumentDB 5.0 unterstützt jetzt Indexscans mit dem `$elemMatch` Operator in der ersten Verschachtelungsebene. Indexscans werden unterstützt, wenn der Filter „Nur Abfragen“ über eine Filterebene verfügt. Sie werden jedoch nicht unterstützt, wenn eine verschachtelte `$elemMatch` Abfrage enthalten ist. `$elemMatch`

Wenn Sie beispielsweise in Amazon DocumentDB 5.0 den `$elemMatch` Operator in die verschachtelte Ebene aufnehmen, wird kein Wert zurückgegeben, wie dies in Amazon DocumentDB 4.0 der Fall ist:

```
db.foo.insert(
[
    {a: {b: 5}},
    {a: {b: [5]}},
    {a: {b: [3, 7]}},
    {a: [{b: 5}]},
    {a: [{b: 3}, {b: 7}]},
    {a: [{b: [5]}]},
    {a: [{b: [3, 7]}]},
    {a: [[{b: 5}]]},
    {a: [[{b: 3}, {b: 7}]]},
    {a: [[{b: [5]}]]},
    {a: [[{b: [3, 7]}]]}
]);

// DocumentDB 5.0
> db.foo.find({a: {$elemMatch: {b: {$elemMatch: {$lt: 6, $gt: 4}}}}}, {_id: 0})
{ "a" : [ { "b" : [ 5 ] } ] }

// DocumentDB 4.0
> db.foo.find({a: {$elemMatch: {b: {$elemMatch: {$lt: 6, $gt: 4}}}}}, {_id: 0})
{ "a" : [ { "b" : [ 5 ] } ] }
{ "a" : [ [ { "b" : [ 5 ] } ] ] }
```

- Die „$“ -Projektion in Amazon DocumentDB 4.0 gibt alle Dokumente mit allen Feldern zurück. In Amazon DocumentDB 5.0 gibt der find Befehl mit einer „$“ -Projektion Dokumente zurück, die dem Abfrageparameter entsprechen und nur das Feld enthalten, das der „$“ -Projektion entspricht.
- In Amazon DocumentDB 5.0 geben die find Befehle mit den `$options` Abfrageparametern `$regex` und den folgenden Fehler zurück: „Optionen können nicht `$regex` sowohl `$options` als auch gesetzt werden“.
- In Amazon DocumentDB 5.0 wird `$indexOfCP` jetzt „-1“ zurückgegeben, wenn:
  - die Teilzeichenfolge wurde im Zeichenfolgenausdruck nicht gefunden, oder
  - Start ist eine Zahl größer als Ende, oder
  - start ist eine Zahl, die größer als die Bytelänge der Zeichenfolge ist.

- Gibt in Amazon DocumentDB 4.0 „0" `$indexOfCP` zurück, wenn die Startposition eine Zahl größer als das Ende oder die Bytelänge der Zeichenfolge ist.
- Bei Amazon DocumentDB 5.0 geben beispielsweise Projektionsoperationen in `_id fields` Dokumente zurück`{"_id.nestedField" : 1}`, die nur das projizierte Feld enthalten. In Amazon DocumentDB 4.0 hingegen filtern Befehle zur Projektion verschachtelter Felder kein Dokument heraus.

MongoDB 4.0-Kompatibilität

Themen

- Funktionen von Amazon DocumentDB 4.0
- Beginnen Sie mit Amazon DocumentDB 4.0
- Führen Sie ein Upgrade durch oder migrieren Sie zu Amazon DocumentDB 4.0
- Funktionelle Unterschiede

Funktionen von Amazon DocumentDB 4.0

Amazon DocumentDB 4.0 führte viele neue Funktionen und Fähigkeiten ein, darunter ACID-Transaktionen und Verbesserungen bei Change-Streams. Die folgende Zusammenfassung zeigt einige der wichtigsten Funktionen, die in Amazon DocumentDB 4.0 eingeführt wurden. Eine vollständige Liste der Funktionen finden Sie imVersionshinweise.

- ACID-Transaktionen: Amazon DocumentDB unterstützt jetzt die Möglichkeit, Transaktionen über mehrere Dokumente, Kontoauszüge, Sammlungen und Datenbanken hinweg durchzuführen. Transaktionen vereinfachen die Anwendungsentwicklung, indem sie es Ihnen ermöglichen, atomare, konsistente, isolierte und dauerhafte (ACID) -Operationen für ein oder mehrere Dokumente innerhalb eines Amazon DocumentDB-Clusters durchzuführen. Weitere Informationen finden Sie unter Transaktionen in Amazon DocumentDB.
- Change Streams: Sie haben jetzt die Möglichkeit, einen Change-Stream auf Cluster-Ebene (`client.watch()`oder`mongo.watch()`) und in der Datenbank (`db.watch()`) zu öffnen, Sie können angeben, dass ein Change-Stream-Cursor geöffnet werden soll, und `startAtOperationTime` zu guter Letzt können Sie jetzt Ihre Aufbewahrungsfrist für Change-Streams auf 7 Tage (zuvor 24 Stunden) verlängern. Weitere Informationen finden Sie unter Change Streams mit Amazon DocumentDB verwenden.

- AWS Database Migration Service(AWS DMS): Sie können es jetzt verwenden, AWS DMS um Ihre MongoDB 4.0-Workloads zu Amazon DocumentDB zu migrieren. AWS DMS unterstützt jetzt eine MongoDB 4.0-Quelle, ein Amazon DocumentDB 4.0-Ziel und eine Amazon DocumentDB 3.6-Quelle für die Durchführung von Upgrades zwischen Amazon DocumentDB 3.6 und 4.0. Weitere Informationen finden Sie in der AWS DMS Dokumentation.
- Leistung und Indizierung: Sie können jetzt einen Index mit verwenden`$lookup`, Abfragen mit einer Projektion finden, die ein Feld oder ein Feld enthalten, und das `_id` Feld kann direkt aus dem Index bereitgestellt werden, ohne dass Sie aus der Sammlung lesen müssen (abgedeckte Abfrage), die Fähigkeit `hint()``findAndModify`, Leistungsoptimierungen für `$addToSet` und Verbesserungen zur Reduzierung der Indexgrößen insgesamt. Weitere Informationen finden Sie unter Versionshinweise.
- Operatoren: Amazon DocumentDB 4.0 unterstützt jetzt eine Reihe neuer Aggregationsoperatoren:`$ifNull`,,,,,`$replaceRoot`. `$setIsSubset` `$setIntersection` `$setUnion` `$setEquals` Alle MongoDB-APIs, Operationen und Datentypen, die wir unterstützen, finden Sie unter. Unterstützte MongoDB-APIs, Operationen und Datentypen in Amazon DocumentDB
- Rollenbasierte Zugriffskontrolle (RBAC): Mit beiden `ListDatabase` Befehlen `ListCollection` und können Sie jetzt optional die `authorizedDatabases` Parameter `authorizedCollections` und verwenden, damit Benutzer die Sammlungen und Datenbanken auflisten können, auf die sie zugreifen dürfen, ohne die jeweiligen `listDatabase` Rollen `listCollections` und zu benötigen. Sie haben auch die Möglichkeit, Ihre eigenen Cursor zu beenden, ohne die Rolle zu benötigen. `KillCursor`

Amazon DocumentDB unterstützt nicht alle MongoDB 4.0-Funktionen. Bei der Entwicklung von Amazon DocumentDB 4.0 haben wir von den Funktionen und Funktionen, um deren Entwicklung uns unsere Kunden am häufigsten gebeten haben, rückwärts gearbeitet. Wir werden weiterhin zusätzliche MongoDB 4.0-Funktionen hinzufügen, je nachdem, was unsere Kunden von uns erwarten. Beispielsweise unterstützt Amazon DocumentDB 4.0 derzeit nicht die Operatoren für die Typkonvertierung oder die Zeichenkettenoperatoren, die in MongoDB 4.0 eingeführt wurden. Die aktuelle Liste der unterstützten APIs finden Sie unter. Unterstützte MongoDB-APIs, Operationen und Datentypen in Amazon DocumentDB

Beginnen Sie mit Amazon DocumentDB 4.0

Informationen zu den ersten Schritten mit Amazon DocumentDB 4.0 finden Sie im. Erste Schritte mit Amazon DocumentDB Sie können einen neuen Amazon DocumentDB 4.0-Cluster mit dem AWS-

Managementkonsole oder dem AWS SDK AWS CLI, oder erstellen. CloudFormation Wenn Sie eine Verbindung zu Amazon DocumentDB herstellen, müssen Sie einen MongoDB-Treiber oder ein Hilfsprogramm verwenden, das mit MongoDB 4.0 oder höher kompatibel ist.

> **Note**
>
> Wenn Sie das AWS SDK oder CloudFormation verwenden AWS CLI, wird die Engine-Version standardmäßig auf 5.0.0 gesetzt. Sie müssen den Parameter explizit angeben`engineVersion = 4.0.0`, um einen neuen Amazon DocumentDB 4.0-Cluster oder einen neuen Amazon DocumentDB 3.6-Cluster `engineVersion = 3.6.0` zu erstellen. Für einen bestimmten Amazon DocumentDB-Cluster können Sie die Cluster-Version ermitteln, indem Sie die Amazon DocumentDB-Managementkonsole aufrufen `describe-db-clusters` oder die Amazon DocumentDB-Managementkonsole verwenden, um die Engine-Versionsnummer für einen bestimmten Cluster anzuzeigen. AWS CLI

Amazon DocumentDB 4.0 unterstütztr5, `r6gt3.medium`, und `t4g.medium` Instance-Typen für Ihre Cluster und ist in allen unterstützten Versionen verfügbar. AWS-Regionen Für die Nutzung von Amazon DocumentDB 4.0 fallen keine zusätzlichen Kosten an. Weitere Informationen zur Preisgestaltung finden Sie unter Preise für Amazon DocumentDB (mit MongoDB-Kompatibilität).

Führen Sie ein Upgrade durch oder migrieren Sie zu Amazon DocumentDB 4.0

Sie können von MongoDB 3.6 oder MongoDB 4.0 zu Amazon DocumentDB 4.0 migrieren, indem Sie Dienstprogramme wie mongodump, mongorestore, mongoimport und mongoexport verwenden. AWS DMS Ebenso können Sie dieselben Tools für ein Upgrade von Amazon DocumentDB 3.6 auf Amazon DocumentDB 4.0 verwenden. Anweisungen zur Migration finden Sie unter. Aktualisieren Ihres Amazon DocumentDB-Clusters mit AWS Database Migration Service

Funktionelle Unterschiede

Funktionale Unterschiede zwischen Amazon DocumentDB 3.6 und 4.0

Mit der Veröffentlichung von Amazon DocumentDB 4.0 gibt es funktionale Unterschiede zwischen Amazon DocumentDB 3.6 und Amazon DocumentDB 4.0:

- Projektion für verschachtelte Dokumente: Amazon DocumentDB 3.6 berücksichtigt das erste Feld in einem verschachtelten Dokument, wenn eine Projektion angewendet wird. Amazon

DocumentDB 4.0 analysiert jedoch Unterdokumente und wendet die Projektion auch auf jedes Unterdokument an. Beispiel: Wenn die Projektion gleich ist`"a.b.c": 1`, dann ist das Verhalten in beiden Versionen identisch. Wenn die Projektion jedoch lautet`{a:{b:{c:1}}}`, wendet Amazon DocumentDB 3.6 die Projektion nur auf 'a' an und nicht auf 'b' oder 'c'.

- Verhalten für**`minKey`**, **`maxKey`**: In Amazon DocumentDB 4.0 gibt das Verhalten für nichts und für alles `{x:{$gt:MaxKey}}` zurück. `{x:{$lt:MaxKey}}`
- Unterschiede beim Dokumentenvergleich: Der Vergleich von numerischen Werten verschiedener Typen (Double, Int, Long) in Unterdokumenten (z. B. b in`{"_id" :1, "a" :{"b":1}}`) sorgt jetzt für eine konsistente Ausgabe für alle numerischen Datentypen und für jede Ebene eines Dokuments.

Funktionale Unterschiede zwischen Amazon DocumentDB 4.0 und MongoDB 4.0

Im Folgenden sind die funktionalen Unterschiede zwischen Amazon DocumentDB 4.0 und MongoDB 4.0 aufgeführt.

- Suche mit leerem Schlüssel im Pfad: Wenn eine Sammlung ein Dokument mit leerem Schlüssel im Array enthält (z. B.`{"x" : [ { "" : 10 }, { "b" : 20 } ]}`) und wenn der in der Abfrage verwendete Schlüssel mit einer leeren Zeichenfolge endet (z. B.`x.`), gibt Amazon DocumentDB dieses Dokument zurück, da es alle Dokumente im Array durchläuft, wohingegen MongoDB dieses Dokument nicht zurückgibt.
- **`$setOnInsert`**zusammen mit **`$`** im Pfad: Der Feldoperator `$setOnInsert` funktioniert nicht in Kombination mit im Pfad $ in Amazon DocumentDB, was auch mit MongoDB 4.0 konsistent ist.

Funktionale Unterschiede: Amazon DocumentDB und MongoDB

Im Folgenden sind die funktionalen Unterschiede zwischen Amazon DocumentDB (mit MongoDB-Kompatibilität) und MongoDB aufgeführt.

Themen

- Funktionale Vorteile von Amazon DocumentDB
- Die funktionalen Unterschiede wurden aktualisiert
- Funktionale Unterschiede zu MongoDB

Funktionale Vorteile von Amazon DocumentDB

Implizite Transaktionen

In Amazon DocumentDB garantieren alle CRUD-Anweisungen (`findAndModify`,, `updateinsert`,`delete`) Atomarität und Konsistenz, selbst bei Vorgängen, bei denen mehrere Dokumente geändert werden. Mit der Einführung von Amazon DocumentDB 4.0 werden nun explizite Transaktionen unterstützt, die ACID-Eigenschaften für Operationen mit mehreren Anweisungen und mehreren Sammlungen bereitstellen. Weitere Informationen zur Verwendung von Transaktionen in Amazon DocumentDB finden Sie unter. Transaktionen in Amazon DocumentDB

Im Folgenden finden Sie Beispiele für Operationen in Amazon DocumentDB, die mehrere Dokumente so modifizieren, dass sie sowohl atomaren als auch konsistenten Verhaltensweisen entsprechen.

```
db.miles.update(
    { "credit_card": { $eq: true } },
    { $mul: { "flight_miles.$[]": NumberInt(2) } },
    { multi: true }
)
```

```
db.miles.updateMany(
    { "credit_card": { $eq: true } },
    { $mul: { "flight_miles.$[]": NumberInt(2) } }
)
```

```
db.runCommand({
  update: "miles",
  updates: [
    {
      q: { "credit_card": { $eq: true } },
      u: { $mul: { "flight_miles.$[]": NumberInt(2) } },
      multi: true
    }
  ]
})
```

```
db.products.deleteMany({
```

```
  "cost": { $gt: 30.00 }
})
```

```
db.runCommand({
  delete: "products",
  deletes: [{ q: { "cost": { $gt: 30.00 } }, limit: 0 }]
})
```

Die einzelnen Operationen, aus denen die Massenoperationen bestehen, wie `updateMany` und `deleteMany`, sind atomar, aber die Gesamtheit der Massenoperation ist nicht atomar. Zum Beispiel ist die Gesamtheit der `insertMany`-Operation atomar, wenn die einzelnen Einfügeoperationen erfolgreich ohne Fehler ausgeführt werden. Tritt bei einer `insertMany` Operation ein Fehler auf, wird jede einzelne Insert-Anweisung innerhalb der `insertMany` Operation als atomare Operation ausgeführt. Wenn Sie ACID-Eigenschaften für `insertMany` `updateMany` ,- und `deleteMany` -Operationen benötigen, wird empfohlen, eine Transaktion zu verwenden.

Die funktionalen Unterschiede wurden aktualisiert

Amazon DocumentDB verbessert weiterhin die Kompatibilität mit MongoDB, indem es ausgehend von den Funktionen, um deren Entwicklung uns unsere Kunden gebeten haben, rückwärts arbeitet. Dieser Abschnitt enthält die funktionalen Unterschiede, die wir in Amazon DocumentDB entfernt haben, um unseren Kunden Migrationen und das Erstellen von Anwendungen zu erleichtern.

Themen

- Array-Indizierung
- Multi-key Indizes
- Nullzeichen in Zeichenketten
- Role-based Zugriffskontrolle
- $regex-Indizierung
- Projektion für verschachtelte Dokumente

Array-Indizierung

Seit dem 23. April 2020 unterstützt Amazon DocumentDB jetzt die Möglichkeit, Arrays zu indizieren, die größer als 2.048 Byte sind. Das Limit für ein einzelnes Element in einem Array liegt weiterhin bei 2.048 Byte, was mit MongoDB konsistent ist.

Beim Erstellen eines neuen Index sind keine Maßnahmen erforderlich, um die Vorteile der verbesserten Funktionalität zu nutzen. Wenn Sie über einen vorhandenen Index verfügen, können Sie die verbesserte Funktionalität nutzen, indem Sie den Index löschen und anschließend neu erstellen. Die aktuelle Index-Version mit den verbesserten Fähigkeiten lautet `"v" : 3`.

> **Note**
>
> Bei Produktionsclustern kann sich das Löschen des Indexes auf die Leistung Ihrer Anwendung auswirken. Testen Sie zuerst und gehen Sie vorsichtig vor, wenn Sie Änderungen an einem Produktionssystem vornehmen. Darüber hinaus hängt die Zeit, die für die Neuerstellung des Indexes benötigt wird, von der Gesamtdatengröße der Sammlung ab.

Mit folgendem Befehl können Sie die Version Ihrer Indizes abfragen.

```
db.collection.getIndexes()
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus. In dieser Ausgabe ist die Indexversion `"v" : 3`, die die aktuellste Indexversion ist.

```
[
    {
        "v" : 3,
        "key" : {
        "_id" : 1
        },
        "name" : "_id_",
        "ns" : "test.test"
    }
]
```

Multi-key Indizes

Seit dem 23. April 2020 unterstützt Amazon DocumentDB jetzt die Möglichkeit, einen zusammengesetzten Index mit mehreren Schlüsseln im selben Array zu erstellen.

Beim Erstellen eines neuen Index sind keine Maßnahmen erforderlich, um die Vorteile der verbesserten Funktionalität zu nutzen. Wenn Sie über einen vorhandenen Index verfügen, können Sie die verbesserte Funktionalität nutzen, indem Sie den Index löschen und anschließend neu erstellen. Die aktuelle Index-Version mit den verbesserten Fähigkeiten lautet `"v" : 3`.

> **Note**
>
> Bei Produktionsclustern kann sich das Löschen des Indexes auf die Leistung Ihrer Anwendung auswirken. Testen Sie zuerst und gehen Sie vorsichtig vor, wenn Sie Änderungen an einem Produktionssystem vornehmen. Darüber hinaus hängt die Zeit, die für die Neuerstellung des Indexes benötigt wird, von der Gesamtdatengröße der Sammlung ab.

Mit folgendem Befehl können Sie die Version Ihrer Indizes abfragen.

```
db.collection.getIndexes()
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus. In dieser Ausgabe ist die Indexversion `"v" : 3`, die die aktuellste Indexversion ist.

```
[
    {
        "v" : 3,
        "key" : {
            "_id" : 1
        },
        "name" : "_id_",
        "ns" : "test.test"
    }
]
```

Nullzeichen in Zeichenketten

Seit dem 22. Juni 2020 unterstützt Amazon DocumentDB jetzt Nullzeichen (`'\0'`) in Zeichenketten.

Role-based Zugriffskontrolle

Seit dem 26. März 2020 unterstützt Amazon DocumentDB die rollenbasierte Zugriffskontrolle (RBAC) für integrierte Rollen. Weitere Informationen hierzu finden Sie unter Role-Based Zugriffskontrolle.

`$regex-Indizierung`

Seit dem 22. Juni 2020 unterstützt Amazon DocumentDB nun die Möglichkeit für `$regex` Betreiber, einen Index zu verwenden.

Um einen Index mit dem `$regex`-Operator zu nutzen, müssen Sie den `hint()`-Befehl verwenden. Bei der Verwendung von `hint()` müssen Sie den Namen des Feldes angeben, auf dem Sie `$regex` anwenden möchten. Wenn Sie beispielsweise einen Index für Feld `product` mit dem Indexnamen `p_1` haben, nutzt `db.foo.find({product: /^x.*/}).hint({product:1})` den `p_1`-Index, aber `db.foo.find({product: /^x.*/}).hint("p_1")` nutzt den Index nicht. Sie können mit dem `explain()`-Befehl überprüfen, ob ein Index ausgewählt wird, oder indem Sie den Profiler zum Protokollieren langsamer Abfragen verwenden. Beispiel, `db.foo.find({product: /^x.*/}).hint("p_1").explain()`.

> **Note**
>
> Die `hint()`-Methode kann nur mit jeweils einem Index verwendet werden.

Die Verwendung eines Index für eine `$regex`-Abfrage ist für regex-Abfragen optimiert, die ein Präfix verwenden und bei denen die regex-Optionen `i`, `m` oder `o` nicht angegeben werden.

Wenn Sie einen Index mit `$regex` verwenden, wird empfohlen, einen Index für hoch selektive Felder zu erstellen, bei denen die Anzahl der doppelten Werte weniger als 1 % der Gesamtzahl der Dokumente in der Sammlung beträgt. Wenn Ihre Sammlung beispielsweise 100.000 Dokumente enthält, erstellen Sie Indizes nur für Felder, in denen derselbe Wert 1.000 Mal oder weniger vorkommt.

Projektion für verschachtelte Dokumente

In Version 3.6 gibt es einen funktionalen Unterschied mit dem `$project` Operator zwischen Amazon DocumentDB und MongoDB, der in Amazon DocumentDB 4.0 behoben wurde, aber in Amazon DocumentDB 3.6 weiterhin nicht unterstützt wird.

Amazon DocumentDB 3.6 berücksichtigt beim Anwenden einer Projektion nur das erste Feld in einem verschachtelten Dokument, wohingegen MongoDB 3.6 Unterdokumente analysiert und die Projektion auch auf jedes Unterdokument anwendet.

Beispiel: Wenn die Projektion zutrifft`"a.b.c": 1`, funktioniert das Verhalten sowohl in Amazon DocumentDB als auch in MongoDB erwartungsgemäß. Wenn die Projektion jedoch `{a:{b:{c:1}}}` so ist, wendet Amazon DocumentDB 3.6 die Projektion nur auf `a` und nicht auf oder an. `b` `c` In Amazon DocumentDB 4.0 `{a:{b:{c:1}}}` wird die Projektion auf `ab`, und angewendet. `c`

Funktionale Unterschiede zu MongoDB

Themen

- $VectorSearch-Operator
- OpCountersCommand
- Admin-Datenbanken und Sammlungen
- Cursor MaxTimeMS
- explain()
- Der Index wird erstellt
- Suche mit leerem Schlüssel im Pfad
- MongoDB-APIs, Operationen und Datentypen
- Dienstprogramme Mongodump und Mongorestore
- Reihenfolge der Ergebnisse
- Wiederholbare Schreibvorgänge
- Sparse Index
- Verwendung von $elemMatch innerhalb eines $all-Ausdrucks
- Dollar ($) und Punkt (.) in Feldnamen
- $lookup
- $natural und umgekehrte Sortierung

$VectorSearch-Operator

Amazon DocumentDB wird `$vectorSearch` als unabhängiger Betreiber nicht unterstützt. Stattdessen unterstützen wir `vectorSearch` innerhalb des `$search` Betreibers. Weitere Informationen finden Sie unter Vektorsuche für Amazon DocumentDB.

`OpCountersCommand`

Das `OpCountersCommand` Verhalten von Amazon DocumentDB weicht wie folgt von dem von MongoDB ab: `opcounters.command`

- MongoDB `opcounters.command` zählt alle Befehle außer Einfügen, Aktualisieren und Löschen, während Amazon DocumentDB den Befehl ebenfalls ausschließt. `OpCountersCommand` `find`
- Amazon DocumentDB zählt einige interne Befehle zu. `OpCountersCommand`

Admin-Datenbanken und Sammlungen

Amazon DocumentDB unterstützt weder die Admin- oder lokale Datenbank noch MongoDB `system.*` bzw. `startup_log` Sammlungen.

`Cursor MaxTimeMS`

`cursor.maxTimeMS`Setzt in Amazon DocumentDB den Zähler für jede Anfrage zurück. `getMore` Wenn also 3000 MS angegeben sind, `maxTimeMS` dauert die Abfrage 2800 MS, und jede nachfolgende `getMore` Anforderung dauert 300 MS, sodass für den Cursor kein Timeout auftritt. Für den Cursor kommt es nur dann zu einer Zeitüberschreitung, wenn eine einzelne Operation, entweder die Abfrage oder eine einzelne `getMore` Anforderung, mehr als die angegebene Zeit in Anspruch nimmt. `maxTimeMS` Außerdem wird der Sweeper, der die Ausführungszeit des Cursors überprüft, mit einer Granularität von fünf (5) Minuten ausgeführt.

explain()

Amazon DocumentDB emuliert die MongoDB-APIs 3.6, 4.0, 5.0 und 8.0 auf einer speziell entwickelten Datenbank-Engine, die ein verteiltes, fehlertolerantes Speichersystem mit Selbstheilung verwendet. Daher können sich die Abfragepläne und die Ausgabe von zwischen Amazon DocumentDB und MongoDB unterscheiden. `explain()` Kunden, die die Kontrolle über ihren Abfrageplan wünschen, können den `$hint`-Operator verwenden, um die Auswahl eines bevorzugten Indexes zu erzwingen.

Der Index wird erstellt

Amazon DocumentDB erlaubt zu einem bestimmten Zeitpunkt nur die Erstellung eines Indexes für eine Sammlung. Entweder im Vordergrund oder im Hintergrund. Wenn Vorgänge wie `createIndex()` oder `dropIndex()` für dieselbe Sammlung auftreten, wenn gerade ein Index erstellt wird,, schlägt der neu versuchte Vorgang fehl.

Standardmäßig werden Index-Builds in Amazon DocumentDB und MongoDB Version 4.0 im Vordergrund ausgeführt. MongoDB Version 4.2 und höher ignoriert die Option zum Erstellen von Indizes im Hintergrund, wenn sie für createIndexes oder seine Shell-Helfer und angegeben wurde. `createIndex()` `createIndexes()`

Ein TTL-Index (Time to Live) beginnt mit dem Ablaufen von Dokumenten, nachdem die Indexerstellung abgeschlossen ist.

Suche mit leerem Schlüssel im Pfad

Wenn Sie mit einem Schlüssel suchen, der eine leere Zeichenfolge als Teil des Pfads enthält (z. B.`x..b`)`x.`, und das Objekt einen leeren Zeichenkettenschlüsselpfad (z. B.`{"x" : [ { "" : 10 }, { "b" : 20 } ]}`) in einem Array hat, gibt Amazon DocumentDB andere Ergebnisse zurück, als wenn Sie dieselbe Suche in MongoDB ausführen würden.

In MongoDB funktioniert die Suche nach einem leeren Schlüsselpfad innerhalb des Arrays wie erwartet, wenn sich der leere Zeichenfolgenschlüssel nicht am Ende der Pfadsuche befindet. Wenn sich der leere Zeichenkettenschlüssel jedoch am Ende der Pfadsuche befindet, wird das Array nicht durchsucht.

In Amazon DocumentDB wird jedoch nur das erste Element innerhalb des Arrays gelesen, da eine leere Zeichenfolge `getArrayIndexFromKeyString` in konvertiert wird`0`, sodass die Suche nach einem Zeichenkettenschlüssel als eine Suche nach einem Array-Index behandelt wird.

MongoDB-APIs, Operationen und Datentypen

Amazon DocumentDB ist mit den MongoDB-APIs 3.6, 4.0, 5.0 und 8.0 kompatibel. Die aktuelle Liste der unterstützten Funktionen finden Sie unter Unterstützte MongoDB-APIs, Operationen und Datentypen in Amazon DocumentDB.

`Dienstprogramme Mongodump` und Mongorestore

Amazon DocumentDB unterstützt keine Admin-Datenbank und speichert die Admin-Datenbank daher nicht und stellt sie auch nicht wieder her, wenn Sie die Dienstprogramme oder verwenden. `mongodump` `mongorestore` Wenn Sie mithilfe von Amazon DocumentDB eine neue Datenbank erstellen`mongorestore`, müssen Sie zusätzlich zum Wiederherstellungsvorgang die Benutzerrollen neu erstellen.

> **Versionsanforderung für die MongoDB-Datenbanktools**
>
> Verwenden Sie die MongoDB-Datenbanktools bis einschließlich Version 100.11.0 für Amazon DocumentDB. Informationen zum Herunterladen finden Sie in den Versionen der MongoDB Database Tools auf der MongoDB-Website.

Reihenfolge der Ergebnisse

Amazon DocumentDB garantiert keine implizite Sortierreihenfolge von Ergebnismengen. Um die Reihenfolge einer Ergebnismenge zu gewährleisten, geben Sie mit `sort()` explizit eine Sortierreihenfolge an.

Das folgende Beispiel sortiert die Artikel in der Inventur in absteigender Reihenfolge basierend auf dem Bestandsfeld.

```
db.inventory.find().sort({ stock: -1 })
```

Bei Verwendung der `$sort` Aggregationsstufe wird die Sortierreihenfolge nicht beibehalten, es sei denn, die `$sort` Phase ist die letzte Phase in der Aggregationspipeline. Wenn die `$sort` Aggregationsstufe in Kombination mit der Aggregationsstufe verwendet wird, wird die `$group` Aggregationsstufe nur auf die `$sort` Akkumulatoren und angewendet. `$first` `$last` In Amazon DocumentDB 4.0 wurde die Unterstützung hinzugefügt, um die Sortierreihenfolge aus der vorherigen Phase `$push` zu berücksichtigen. `$sort`

Wiederholbare Schreibvorgänge

Die meisten aktuellen MongoDB-compatible Treiber ermöglichen standardmäßig wiederholbare Schreibvorgänge. Amazon DocumentDB unterstützt ab der Engine-Version 8.0.2 wiederholbare Schreibvorgänge. In früheren Engine-Versionen werden wiederholbare Schreibvorgänge nicht unterstützt und Sie müssen sie deaktivieren, um Fehler zu vermeiden.

Bei Engine-Versionen vor 8.0.2 führt die Aktivierung wiederholbarer Schreibversuche zu einer Fehlermeldung ähnlich der folgenden.

```
{"ok":0,"errmsg":"Retryable writes are not supported","code":301,"name":"MongoError"}
```

Wiederholbare Schreibvorgänge können über die Verbindungszeichenfolge (z. B.`MongoClient("mongodb://my.mongodb.cluster/db?retryWrites=false")`) oder das Schlüsselwortargument des MongoClient Konstruktors (z. B.) deaktiviert werden. `MongoClient("mongodb://my.mongodb.cluster/db", retryWrites=False)`

Weitere Hinweise zu wiederholbaren Schreibvorgängen auf Engine-Version 8.0.2 und höher finden Sie unter. Wiederholbare Schreibvorgänge in Amazon DocumentDB

Sparse Index

Um einen Sparse-Index zu verwenden, den Sie in einer Abfrage erstellt haben, müssen Sie die `$exists`-Bedingungen für die Felder verwenden, die der Index abdecken. Wenn Sie dies weglassen`$exists`, verwendet Amazon DocumentDB den Index mit geringer Dichte nicht.

Im Folgenden wird ein -Beispiel gezeigt.

```
db.inventory.count({ "stock": { $exists: true }})
```

Bei Indizes mit geringer Dichte und mehreren Schlüsseln unterstützt Amazon DocumentDB keine eindeutige Schlüsseleinschränkung, wenn die Suche nach einem Dokument zu einer Reihe von Werten führt und nur eine Teilmenge der indizierten Felder fehlt. Beispielsweise wird `createIndex({"a.b" : 1 }, { unique : true, sparse :true })` nicht unterstützt, wenn `"a" : [ { "b" : 2 }, { "c" : 1 } ]` eingegeben wird, da `"a.c"` im Index gespeichert ist.

Verwendung von $elemMatch innerhalb eines $all-Ausdrucks

Amazon DocumentDB unterstützt derzeit nicht die Verwendung des `$elemMatch` Operators innerhalb eines Ausdrucks. `$all` Sie können dies umgehen, indem Sie den Operator `$and` wie folgt mit `$elemMatch` verwenden.

Ursprüngliche Operation:

```
db.col.find({
  qty: {
    $all: [
      { "$elemMatch": { part: "xyz", qty: { $lt: 11 } } },
      { "$elemMatch": { num: 40, size: "XL" } }
    ]
  }
})
```

Aktualisierte Operation:

```
db.col.find({
  $and: [
    { qty: { "$elemMatch": { part: "xyz", qty: { $lt: 11 } } } },
    { qty: { "$elemMatch": { qty: 40, size: "XL" } } }
  ]
})
```

Dollar ($) und Punkt (.) in Feldnamen

Amazon DocumentDB unterstützt nicht die Abfrage von Feldern mit dem Präfix Dollar ($) in $in, $nin und $all in verschachtelten Objekten. Beispielsweise ist die folgende Abfrage in Amazon DocumentDB nicht gültig:

```
coll.find({"field": {"$all": [{ "$a": 1 }]}})
```

`$lookup`

Amazon DocumentDB unterstützt die Möglichkeit, Gleichheitsübereinstimmungen durchzuführen (z. B. Left Outer Join) und unterstützt auch unkorrelierte Unterabfragen. Ab Version 8.0.2 unterstützt Amazon DocumentDB auch korrelierte Unterabfragen, einschließlich der präzisen korrelierten Unterabfragesyntax. In einer korrelierten Unterabfrage kann nur eine Teilmenge der Aggregationsstufen, die Ausdrücke auswerten, auf die Variablen verweisen. `let` Eine Liste der unterstützten Stufen und der zugehörigen Einschränkungen finden Sie unter. Unterstützte Stufen in korrelierten Unterabfragen In Versionen vor 8.0.2 werden korrelierte Unterabfragen nicht unterstützt.

Verwendung eines Indexes mit $lookup

Sie können jetzt einen Index mit dem `$lookup` Stage-Operator verwenden. Je nach Anwendungsfall gibt es mehrere Indizierungsalgorithmen, mit denen Sie die Leistung optimieren können. In diesem Abschnitt werden die verschiedenen Indexierungsalgorithmen für Ihre Arbeitslast erläutert `$lookup` und Sie bei der Auswahl des besten Algorithmus unterstützt.

Standardmäßig verwendet Amazon DocumentDB den Hash-Algorithmus, wenn `allowDiskUse:false` er verwendet wird, und Sort Merge, wenn er verwendet `allowDiskUse:true` wird.

> **Note**
>
> Die `allowDiskUse` Option wird derzeit für den `find` Befehl nicht unterstützt. Die Option wird nur als Teil der Aggregation unterstützt. Verwenden Sie das Aggregationsframework mit`allowDiskUse:true`, um große Abfragen zu verarbeiten, die die Speichergrenzen überschreiten könnten.

In einigen Anwendungsfällen kann es wünschenswert sein, den Abfrageoptimierer zu zwingen, einen anderen Algorithmus zu verwenden. Im Folgenden sind die verschiedenen Indizierungsalgorithmen aufgeführt, die der `$lookup` Aggregationsoperator verwenden kann:

- Verschachtelte Schleife: Ein Plan für verschachtelte Schleifen ist in der Regel für eine Arbeitslast von Vorteil, wenn die ausländische Sammlung <1 GB groß ist und das Feld in der fremden Sammlung einen Index hat. Wenn der Nested-Loop-Algorithmus verwendet wird, zeigt der Explain-Plan die Phase als. `NESTED_LOOP_LOOKUP`
- Sortierzusammenführung: Ein Plan zum Zusammenführen von Sortierungen ist in der Regel für eine Arbeitslast von Vorteil, wenn die ausländische Sammlung keinen Index für das bei der Suche verwendete Feld hat und der Arbeitsdatensatz nicht in den Arbeitsspeicher passt. Wenn der Algorithmus zum Zusammenführen von Sortierungen verwendet wird, wird die Phase im Explain-Plan als angezeigt`SORT_LOOKUP`.
- Hash: Ein Hash-Plan ist in der Regel für eine Arbeitslast von Vorteil, wenn die ausländische Sammlung < 1 GB groß ist und der Arbeitsdatensatz in den Arbeitsspeicher passt. Wenn der Hash-Algorithmus verwendet wird, wird die Phase im Explain-Plan als angezeigt`HASH_LOOKUP`.

Sie können den Indexierungsalgorithmus, der für den `$lookup` Operator verwendet wird, `explain` anhand der Abfrage identifizieren. Im Folgenden finden Sie ein Beispiel:

```
db.localCollection.explain().aggregate(
   [
      {
         $lookup:
            {
               from: "foreignCollection",
               localField: "a",
               foreignField: "b",
               as: "joined"
```

```
            }
        }
    ]
)

output
{
    "queryPlanner" : {
        "plannerVersion" : 1,
        "namespace" : "test.localCollection",
        "winningPlan" : {
            "stage" : "SUBSCAN",
            "inputStage" : {
                "stage" : "SORT_AGGREGATE",
                "inputStage" : {
                    "stage" : "SORT",
                    "inputStage" : {
                        "stage" : "NESTED_LOOP_LOOKUP",
                        "inputStages" : [
                            {
                                "stage" : "COLLSCAN"
                            },
                            {
                                "stage" : "FETCH",
                                "inputStage" : {
                                    "stage" : "COLLSCAN"
                                }
                            }
                        ]
                    }
                }
            }
        }
    },
    "serverInfo" : {
        "host" : "devbox-test",
        "port" : 27317,
        "version" : "3.6.0"
    },
    "ok" : 1
}
```

Als Alternative zur Verwendung der `explain()` Methode können Sie den Profiler verwenden, um den Algorithmus zu überprüfen, der bei Ihrer Verwendung des `$lookup` Operators verwendet wird. Weitere Informationen zum Profiler finden Sie unter. Profilierung von Amazon DocumentDB-Vorgängen

Einen PlanHint **verwenden**

Wenn Sie den Abfrageoptimierer zwingen möchten, einen anderen Indexierungsalgorithmus mit zu verwenden`$lookup`, können Sie einen verwenden. `planHint` Verwenden Sie dazu den Kommentar in den Optionen der Aggregationsphase, um einen anderen Plan zu erzwingen. Unten finden Sie ein Beispiel für die Syntax für den Kommentar:

```
comment : {
    comment :  "<string>",
    lookupStage : { planHint : "SORT" | "HASH" | "NESTED_LOOP" }
}
```

Im Folgenden finden Sie ein Beispiel für die Verwendung von`planHint`, um den Abfrageoptimierer zur Verwendung des HASH Indexierungsalgorithmus zu zwingen:

```
db.foo.aggregate(
   [
      {
         $lookup:
            {
               from: "foo",
               localField: "_id",
               foreignField: "_id",
               as: "joined"
            },
      }
   ]
),
{
   comment : "{ \"lookupStage\" : { \"planHint\": \"HASH\" }}"
```

Um zu testen, welcher Algorithmus für Ihre Arbeitslast am besten geeignet ist, können Sie den `executionStats` Parameter der `explain` Methode verwenden, um die Ausführungszeit der `$lookup` Phase zu messen und gleichzeitig den Indexierungsalgorithmus zu ändern (d. h.HASH//SORT). NESTED_LOOP

Das folgende Beispiel zeigt, wie Sie `executionStats` die Ausführungszeit der `$lookup` Stufe mithilfe des `SORT` Algorithmus messen können.

```
db.foo.explain("executionStats").aggregate(
   [
      {
         $lookup:
            {
               from: "foo",
               localField: "_id",
               foreignField: "_id",
               as: "joined"
            },
      }
   ]
),
{
   comment : "{ \"lookupStage\" : { \"planHint\": \"SORT\" }}"
```

`$natural` und umgekehrte Sortierung

Amazon DocumentDB unterstützt nur `$natural` Forward-Collection-Scans. Scans in umgekehrter Reihenfolge (`{$natural: -1}`) führen zu einem`MongoServerError`.

Unterstützte MongoDB-APIs, Operationen und Datentypen in Amazon DocumentDB

Amazon DocumentDB (mit MongoDB-Kompatibilität) ist ein schneller, skalierbarer, hochverfügbarer und vollständig verwalteter Dokumentdatenbankdienst, der MongoDB-Workloads unterstützt. Amazon DocumentDB ist mit den MongoDB-APIs 3.6, 4.0, 5.0 und 8.0 kompatibel. In diesem Abschnitt werden die unterstützten Funktionalitäten aufgeführt. Unterstützung bei der Verwendung von MongoDB-APIs und -Treibern finden Sie in den MongoDB-Community-Foren. Wenn Sie Unterstützung bei der Nutzung des Amazon DocumentDB-Service benötigen, wenden Sie sich an das entsprechende AWS Support-Team. Informationen zu funktionalen Unterschieden zwischen Amazon DocumentDB und MongoDB finden Sie unter. Funktionale Unterschiede: Amazon DocumentDB und MongoDB

MongoDB-Befehle und -Operatoren, die nur intern oder nicht auf einen verwalteten Service anwendbar sind, werden nicht unterstützt und sind nicht in der Liste der unterstützten Funktionalität enthalten.

Seit der Markteinführung haben wir mehr als 50 zusätzliche Funktionen hinzugefügt und werden uns weiter an den Anforderungen unserer Kunden ausrichten, um ihnen die benötigten Funktionen bereitzustellen. Informationen zu den neuesten Produkteinführungen finden Sie unter Amazon DocumentDB-Ankündigungen.

Wenn es eine Funktion gibt, die nicht unterstützt wird und die Sie von uns erstellen lassen möchten, teilen Sie uns dies mit, indem Sie eine E-Mail mit Ihrer Konto-ID, den angeforderten Funktionen und dem Anwendungsfall an das Amazon DocumentDB-Serviceteam senden.

Topics

- Datenbank-Befehle
- Abfrage- und Projektionsoperatoren
- Operatoren aktualisieren
- Geodaten
- Cursor-Methoden
- Betreiber von Aggregations-Pipelines
- Datentypen
- Indizes

Datenbank-Befehle

Themen

- Administrative Befehle
- Aggregation
- Authentifizierung
- Diagnosebefehle
- Abfrage- und Schreibvorgänge
- Befehle zur Rollenverwaltung
- Befehle für Sitzungen

- Benutzerverwaltung
- Sharding-Befehle

Administrative Befehle

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
Gedeckelte Sammlungen	Nein	Nein	Nein	Nein	Nein
klonen CollectionAsCapped	Nein	Nein	Nein	Nein	Nein
collMod	Teilweise	Teilweise	Teilweise	Teilweise	Teilweise
CollMod: ablaufenAfterSeconds	Ja	Ja	Ja	Ja	Ja
CollMod: versteckt	Nein	Nein	Ja (5.0.2+)	Ja (8.0.1+)	Nein
konvertierenToCapped	Nein	Nein	Nein	Nein	Nein
copydb	Nein	Nein	Nein	Nein	Nein
Erstellen	Ja	Ja	Ja	Ja	Ja
createView	Nein	Nein	Nein	Ja	Nein
createIndexes	Ja	Ja	Ja	Ja	Ja
currentOp	Ja	Ja	Ja	Ja	Ja
fallen lassen	Ja	Ja	Ja	Ja	Ja

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
dropDatabase	Ja	Ja	Ja	Ja	Ja
dropIndexes	Ja	Ja	Ja	Ja	Ja
filemd5	Nein	Nein	Nein	Nein	Nein
erhaltenAuditConfig	Nein	Ja	Ja	Ja	Nein
killCursors	Ja	Ja	Ja	Ja	Ja
killOp	Ja	Ja	Ja	Ja	Ja
Sammlungenauflisten*	Ja	Ja	Ja	Ja	Ja
listDatabases	Ja	Ja	Ja	Ja	Ja
listIndexes	Ja	Ja	Ja	Ja	Ja
reIndex	Nein	Nein	Ja	Ja	Nein
renameCollection	Ja	Ja	Ja	Ja	Nein
setzenAuditConfig	Nein	Ja	Ja	Ja	Nein

* Der `type` Schlüssel in der Filteroption wird nicht unterstützt.

Aggregation

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
aggregate	Ja	Ja	Ja	Ja	Ja
count	Ja	Ja	Ja	Ja	Ja
distinct	Ja	Ja	Ja	Ja	Ja
mapReduce	Nein	Nein	Nein	Ja	Nein

Authentifizierung

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
authenticate	Ja	Ja	Ja	Ja	Ja
logout	Ja	Ja	Ja	Ja	Ja

Diagnosebefehle

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
buildInfo	Ja	Ja	Ja	Ja	Ja
collStats	Ja	Ja	Ja	Ja	Ja
connPoolStats	Nein	Nein	Nein	Nein	Nein
connectionStatus	Ja	Ja	Ja	Ja	Ja

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
dataSize	Ja	Ja	Ja	Ja	Ja
dbHash	Nein	Nein	Nein	Nein	Nein
dbStats	Ja	Ja	Ja	Ja	Ja
explain	Ja	Ja	Ja	Ja	Ja
Erklären: executionStats	Ja	Ja	Ja	Ja	Ja
features	Nein	Nein	Nein	Nein	Nein
hostInfo	Ja	Ja	Ja	Ja	Ja
listCommands	Ja	Ja	Ja	Ja	Ja
planenCacheClearFilters	Nein	Nein	Ja	Ja	Nein
planenCacheListFilters	Nein	Nein	Ja	Ja	Nein
planenCacheSetFilter	Nein	Nein	Ja	Ja	Nein
Profiler	Ja	Ja	Ja	Ja	Nein
serverStatus	Ja	Ja	Ja	Ja	Ja
top	Ja	Ja	Ja	Ja	Ja

Abfrage- und Schreibvorgänge

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
Change streams	Ja	Ja	Ja	Ja	Nein
delete	Ja	Ja	Ja	Ja	Ja
find	Ja	Ja	Ja	Ja	Ja
finden AndModify	Ja	Ja	Ja	Ja	Ja
bekommen LastError	Nein	Nein	Nein	Nein	Nein
getMore	Ja	Ja	Ja	Ja	Ja
bekommen PrevError	Nein	Nein	Nein	Nein	Nein
GridFS	Ja	Ja	Ja	Ja	Nein
insert	Ja	Ja	Ja	Ja	Ja
parallel CollectionScan	Nein	Nein	Nein	Nein	Nein
resetError	Nein	Nein	Nein	Nein	Nein
update	Ja	Ja	Ja	Ja	Ja
ReplaceOne	Ja	Ja	Ja	Ja	Ja

Befehle zur Rollenverwaltung

E	3.6	4.0	5.0	Elastischer Cluster
c	Ja	Ja	Ja	Nein
f	Ja	Ja	Ja	Nein
d	Ja	Ja	Ja	Nein
g	Ja	Ja	Ja	Nein
v	Ja	Ja	Ja	Nein
v	Ja	Ja	Ja	Nein
r	Ja	Ja	Ja	Nein
u	Ja	Ja	Ja	Nein

Befehle für Sitzungen

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
Transaktion abbrechen	Nein	Ja	Ja	Ja	Nein
commitTransaction	Nein	Ja	Ja	Ja	Nein
Sitzungen beenden	Nein	Nein	Nein	Nein	Nein
töten AllSessions	Nein	Ja	Ja	Ja	Nein
töten AllSessionsByPattern	Nein	Nein	Nein	Nein	Nein
Sitzungen beenden	Nein	Ja	Ja	Ja	Nein
Sitzungen aktualisieren	Nein	Nein	Nein	Nein	Nein
Sitzung starten	Nein	Ja	Ja	Ja	Nein

Benutzerverwaltung

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
createUser	Ja	Ja	Ja	Ja	Ja

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
fallen AllUsersFromDatabase	Ja	Ja	Ja	Ja	Ja
dropUser	Ja	Ja	Ja	Ja	Ja
gewähren RolesToUser	Ja	Ja	Ja	Ja	Ja
widerrufen RolesFrom User	Ja	Ja	Ja	Ja	Ja
updateUser	Ja	Ja	Ja	Ja	Ja
Informationen über Benutzer	Ja	Ja	Ja	Ja	Ja

Sharding-Befehle

Befehl	Elastischer Cluster
abbrechen ReshardCollection	Nein
Shard hinzufügen	Nein
hinzufügen ShardToZone	Nein
Balancer CollectionStatus	Nein
Balancer starten	Nein
Status des Balancers	Nein
Balancer-Stopp	Nein

Befehl	Elastischer Cluster
überprüfen ShardingIndex	Nein
klar JumboFlag	Nein
Räum verwaist auf	Nein
aufräumen ReshardCollection	Nein
begehen ReshardCollection	Nein
Sharding aktivieren	Ja
spülen RouterConfig	Nein
bekommen ShardMap	Nein
bekommen ShardVersion	Nein
isdbgrid	Nein
Listet Shards auf	Nein
Mittlerer Schlüssel	Nein
Chunk verschieben	Nein
Primär verschieben	Nein
Chunks zusammenführen	Nein
verfeinern CollectionShardKey	Nein
Schwer entfernen	Nein
entfernen ShardFromZone	Nein
Sammlung Reshard	Nein
einstellen AllowMigrations	Nein

Befehl	Elastischer Cluster
einstellen ShardVersion	Nein
Sammlung teilen	Ja
Sharding State	Nein
split	Nein
Geteilter Vektor	Nein
Deaktivieren Sie Sharding	Nein
aktualisieren ZoneKeyRange	Nein

Abfrage- und Projektionsoperatoren

Themen

- Array-Operatoren
- Bitweise Operatoren
- Operator „Kommentar“
- Vergleichsoperatoren
- Operatoren für Elemente
- Operatoren für Evaluierungsabfragen
- Logische Operatoren
- Operatoren für Projektionen

Array-Operatoren

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$all	Ja	Ja	Ja	Ja	Ja
$elemMatch	Ja	Ja	Ja	Ja	Ja

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$size	Ja	Ja	Ja	Ja	Ja

Bitweise Operatoren

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$ Bits AllSet	Ja	Ja	Ja	Ja	Ja
$ Bits AnySet	Ja	Ja	Ja	Ja	Ja
$ Bits AllClear	Ja	Ja	Ja	Ja	Ja
$ Bits AnyClear	Ja	Ja	Ja	Ja	Ja

Operator „Kommentar“

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$comment	Ja	Ja	Ja	Ja	Ja

Vergleichsoperatoren

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$eq	Ja	Ja	Ja	Ja	Ja
$gt	Ja	Ja	Ja	Ja	Ja

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$gte	Ja	Ja	Ja	Ja	Ja
$in	Ja	Ja	Ja	Ja	Ja
$lt	Ja	Ja	Ja	Ja	Ja
$lte	Ja	Ja	Ja	Ja	Ja
$ne	Ja	Ja	Ja	Ja	Ja
$nin	Ja	Ja	Ja	Ja	Ja

Operatoren für Elemente

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$exists	Ja	Ja	Ja	Ja	Ja
$type	Ja	Ja	Ja	Ja	Ja

Operatoren für Evaluierungsabfragen

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$expr	Nein	Ja	Ja	Ja	Nein
$jsonSchema	Nein	Ja	Ja	Ja	Nein
$mod	Ja	Ja	Ja	Ja	Ja
$regex	Ja	Ja	Ja	Ja	Ja
$text	Nein	Nein	Ja	Ja	Nein

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$where	Nein	Nein	Nein	Nein	Nein

Logische Operatoren

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$and	Ja	Ja	Ja	Ja	Ja
$nor	Ja	Ja	Ja	Ja	Ja
$not	Ja	Ja	Ja	Ja	Ja
$or	Ja	Ja	Ja	Ja	Ja

Operatoren für Projektionen

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$	Ja	Ja	Ja	Ja	Ja
$elemMatch	Ja	Ja	Ja	Ja	Ja
$meta	Nein	Nein	Ja	Ja	Nein
$slice	Ja	Ja	Ja	Ja	Ja

Operatoren aktualisieren

Themen

- Array-Operatoren
- Bitweise Operatoren

- Operatoren vor Ort
- Modifikatoren aktualisieren

Array-Operatoren

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$	Ja	Ja	Ja	Ja	Ja
$[]	Ja	Ja	Ja	Ja	Ja
$ [<Kennung>]	Ja	Ja	Ja	Ja	Ja
$addToSet	Ja	Ja	Ja	Ja	Ja
$pop	Ja	Ja	Ja	Ja	Ja
$pullAll	Ja	Ja	Ja	Ja	Ja
$pull	Ja	Ja	Ja	Ja	Ja
$push	Ja	Ja	Ja	Ja	Ja

Bitweise Operatoren

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$bit	Ja	Ja	Ja	Ja	Ja

Operatoren vor Ort

Operator	3.6	4,0	5.0	8.0	Elastischer Cluster
$currentDate	Ja	Ja	Ja	Ja	Ja
$inc	Ja	Ja	Ja	Ja	Ja
$max	Ja	Ja	Ja	Ja	Ja
$min	Ja	Ja	Ja	Ja	Ja
$mul	Ja	Ja	Ja	Ja	Ja
$rename	Ja	Ja	Ja	Ja	Ja
$set	Ja	Ja	Ja	Ja	Ja
$setOnInsert	Ja	Ja	Ja	Ja	Ja
$unset	Ja	Ja	Ja	Ja	Ja

Modifikatoren aktualisieren

Operator	3.6	4,0	5.0	8.0	Elastischer Cluster
$each	Ja	Ja	Ja	Ja	Ja
$position	Ja	Ja	Ja	Ja	Ja
$slice	Ja	Ja	Ja	Ja	Ja
$sort	Ja	Ja	Ja	Ja	Ja

Geodaten

Geometrie-Spezifizierer

Abfrageauswahl	3.6	4,0	5.0	8.0	Elastischer Cluster
$box	Nein	Nein	Nein	Nein	Nein
$center	Nein	Nein	Nein	Nein	Nein
$centerSphere	Nein	Nein	Nein	Nein	Nein
$geometry	Ja	Ja	Ja	Ja	Ja
$maxDistance	Ja	Ja	Ja	Ja	Ja
$minDistance	Ja	Ja	Ja	Ja	Ja
$nearSphere	Ja	Ja	Ja	Ja	Ja
$polygon	Nein	Nein	Nein	Nein	Nein
$uniqueDocs	Nein	Nein	Nein	Nein	Nein

Selektoren abfragen

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$geoIntersects	Ja	Ja	Ja	Ja	Ja
$geoWithin	Ja	Ja	Ja	Ja	Ja
$near	Ja	Ja	Ja	Ja	Ja

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$nearSphere	Ja	Ja	Ja	Ja	Ja
$polygon	Nein	Nein	Nein	Nein	Nein
$uniqueDocs	Nein	Nein	Nein	Nein	Nein

Cursor-Methoden

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
cursor.batchSize()	Ja	Ja	Ja	Ja	Ja
cursor.close()	Ja	Ja	Ja	Ja	Ja
cursor.collation()	Nein	Nein	Nein	Ja	Nein
cursor.comment()	Ja	Ja	Ja	Ja	Ja
cursor.count()	Ja	Ja	Ja	Ja	Ja
cursor.explain()	Ja	Ja	Ja	Ja	Nein
cursor.forEach()	Ja	Ja	Ja	Ja	Ja
cursor.hasNext()	Ja	Ja	Ja	Ja	Ja
cursor.hint()	Ja	Ja	Ja	Ja	Ja*

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
cursor.isClosed()	Ja	Ja	Ja	Ja	Ja
cursor.isExhausted()	Ja	Ja	Ja	Ja	Nein
cursor.itcount()	Ja	Ja	Ja	Ja	Nein
cursor.limit()	Ja	Ja	Ja	Ja	Nein
cursor.map()	Ja	Ja	Ja	Ja	Nein
cursor.max()	Nein	Nein	Ja (5.0.2+)	Ja (8.0.1+)	Nein
cursor.maxScan()	Ja	Ja	Ja	Ja	Nein
cursor.maxTimeMS()	Ja	Ja	Ja	Ja	Nein
cursor.min()	Nein	Nein	Ja (5.0.2+)	Ja (8.0.1+)	Nein
cursor.next()	Ja	Ja	Ja	Ja	Ja
cursor.nein() CursorTimeout	Nein	Nein	Nein	Nein	Nein
cursor.objs () LeftInBatch	Ja	Ja	Ja	Ja	Nein
cursor.pretty()	Ja	Ja	Ja	Ja	Nein
cursor.readConcern()	Ja	Ja	Ja	Ja	Nein

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
cursor.readPref()	Ja	Ja	Ja	Ja	Nein
cursor.returnKey()	Nein	Nein	Nein	Nein	Nein
cursor.showRecordId ()	Nein	Nein	Nein	Nein	Nein
cursor.size()	Ja	Ja	Ja	Ja	Nein
cursor.skip()	Ja	Ja	Ja	Ja	Nein
cursor.sort()	Ja	Ja	Ja	Ja	Nein
cursor.tailable()	Nein	Nein	Nein	Nein	Nein
cursor.toArray()	Ja	Ja	Ja	Ja	Nein

* Index `hint` wird mit Indexausdrücken unterstützt. Beispiel, `db.foo.find().hint({x:1})`.

Die `cursor.max()` Methoden `cursor.min()` und benötigen einen`hint`, der den zu verwendenden Index benennt. Beide Methoden müssen dieselben Felder angeben, und Sie können sie nicht mit einer `$natural` Sortierung kombinieren. Sie werden nur für den `find` Befehl unterstützt.

Betreiber von Aggregations-Pipelines

Themen

- Akkumulator-Ausdrücke
- Arithmetische Operatoren
- Array-Operatoren
- Boolesche Operatoren

- Vergleichsoperatoren
- Operatoren für bedingte Ausdrücke
- Datentyp-Operator
- Operator „Datengröße“
- Datumsoperatoren
- Wörtlicher Operator
- Operator zusammenführen
- Natürlicher Operator
- Satzoperatoren
- Operatoren auf der Bühne
- Zeichenfolgenoperatoren
- Systemvariablen
- Operator für die Textsuche
- Operatoren für die Typkonvertierung
- Variable Operatoren
- Trigonometrie-Operatoren
- Bitweise Operatoren (Aggregation)
- Zeitstempel-Operatoren
- Verschiedene Betreiber

Akkumulator-Ausdrücke

Expression	3.6	4,0	5.0	8.0	Elastischer Cluster
$ Akkumulator	-	-	Nein	Nein	Nein
$hinzufügenToSet	Ja	Ja	Ja	Ja	Ja
$avg	Ja	Ja	Ja	Ja	Ja

Expression	3.6	4,0	5.0	8.0	Elastischer Cluster
$ unten	-	-	-	Ja (8.0.1+)	Nein
$ unten N	-	-	-	Ja (8.0.1+)	Nein
$count	-	-	Nein	Ja (8.0.1+)	Nein
$ CovariancePop	Nein	Nein	Nein	Ja (8.0.2+)	Nein
$KovarianzSAMP	Nein	Nein	Nein	Ja (8.0.2+)	Nein
$ DichterRang	Nein	Nein	Nein	Ja (8.0.2+)	Nein
$ Derivat	Nein	Nein	Nein	Nein	Nein
$ Dokumentnummer	Nein	Nein	Nein	Ja (8.0.2+)	Nein
$expMovingAvg	Nein	Nein	Nein	Nein	Nein
$first	Ja	Ja	Ja	Ja	Ja
$FirstN	-	-	-	Ja (8.0.1+)	Nein
$integral	Nein	Nein	Nein	Nein	Nein
$last	Ja	Ja	Ja	Ja	Ja
$ letztes N	-	-	-	Ja (8.0.1+)	Nein
$ linearFill	Nein	Nein	Nein	Nein	Nein
$ locf	Nein	Nein	Nein	Nein	Nein
$max	Ja	Ja	Ja	Ja	Ja

Expression	3.6	4,0	5.0	8.0	Elastischer Cluster
$ MaxN	-	-	-	Ja (8.0.1+)	Nein
$ Median	-	-	-	Ja (8.0.1+)	Nein
$min	Ja	Ja	Ja	Ja	Ja
$ min	-	-	-	Ja (8.0.1+)	Nein
$ Perzentil	-	-	-	Ja (8.0.1+)	Nein
$push	Ja	Ja	Ja	Ja	Ja
$ Rang	Nein	Nein	Nein	Ja (8.0.2+)	Nein
$shift	Nein	Nein	Nein	Nein	Nein
$ std DevPop	Nein	Nein	Nein	Ja (8.0.1+)	Nein
$ std DevSamp	Nein	Nein	Nein	Ja (8.0.1+)	Nein
$sum	Ja	Ja	Ja	Ja	Ja
$top	-	-	-	Ja (8.0.1+)	Nein
$ zu PN	-	-	-	Ja (8.0.1+)	Nein

Arithmetische Operatoren

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$abs	Ja	Ja	Ja	Ja	Ja
$add	Ja	Ja	Ja	Ja	Ja
$ceil	Nein	Ja	Ja	Ja	Ja

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$divide	Ja	Ja	Ja	Ja	Ja
$exp	Nein	Ja	Ja	Ja	Ja
$floor	Nein	Ja	Ja	Ja	Ja
$ln	Nein	Ja	Ja	Ja	Ja
$log	Nein	Ja	Ja	Ja	Ja
$log10	Nein	Ja	Ja	Ja	Ja
$mod	Ja	Ja	Ja	Ja	Ja
$multiply	Ja	Ja	Ja	Ja	Ja
$pow	Nein	Nein	Ja (5.0.1+)	Ja	Nein
$ Runde	-	-	Nein	Ja (8.0.1+)	Nein
$sqrt	Nein	Ja	Ja	Ja	Ja
$subtract	Ja	Ja	Ja	Ja	Ja
$trunc	Nein	Nein	Nein	Ja (8.0.1+)	Nein

Array-Operatoren

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$arrayElemAt	Ja	Ja	Ja	Ja	Ja
$arrayToObject	Ja	Ja	Ja	Ja	Ja

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$concatArrays	Ja	Ja	Ja	Ja	Ja
$filter	Ja	Ja	Ja	Ja	Ja
$first	-	-	Ja	Ja	Nein
$in	Ja	Ja	Ja	Ja	Ja
$ indexOfArray	Ja	Ja	Ja	Ja	Ja
$isArray	Ja	Ja	Ja	Ja	Ja
$last	-	-	Ja	Ja	Nein
$objektToArray	Ja	Ja	Ja	Ja	Ja
$range	Ja	Ja	Ja	Ja	Ja
$reverseArray	Ja	Ja	Ja	Ja	Ja
$sortArray	Nein	Nein	Nein	Ja	Nein
$reduce	Ja	Ja	Ja	Ja	Ja
$size	Ja	Ja	Ja	Ja	Ja
$slice	Ja	Ja	Ja	Ja	Ja
$zip	Ja	Ja	Ja	Ja	Ja

Boolesche Operatoren

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$and	Ja	Ja	Ja	Ja	Ja
$not	Ja	Ja	Ja	Ja	Ja
$or	Ja	Ja	Ja	Ja	Ja

Vergleichsoperatoren

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$cmp	Ja	Ja	Ja	Ja	Ja
$eq	Ja	Ja	Ja	Ja	Ja
$gt	Ja	Ja	Ja	Ja	Ja
$gte	Ja	Ja	Ja	Ja	Ja
$lt	Ja	Ja	Ja	Ja	Ja
$lte	Ja	Ja	Ja	Ja	Ja
$ne	Ja	Ja	Ja	Ja	Ja

Operatoren für bedingte Ausdrücke

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$cond	Ja	Ja	Ja	Ja	Ja

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$ifNull	Ja	Ja	Ja	Ja	Ja
$switch	Nein	Ja	Ja	Ja	Nein

Datentyp-Operator

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$type	Ja	Ja	Ja	Ja	Ja

Operator „Datengröße“

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$ Binärgröße	-	-	Nein	Ja (8.0.1+)	Nein
$ bsonGröße	-	-	Nein	Ja (8.0.1+)	Nein

Datumsoperatoren

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$DateAdd	Nein	Nein	Ja	Ja	Ja
$DateDiff	-	-	Ja	Ja	Nein
$datum FromParts	Nein	Nein	Ja (5.0.1+)	Nein	Nein

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$datumFromString	Ja	Ja	Ja	Ja	Ja
$DateSubtrahieren	Nein	Nein	Ja	Ja	Ja
$datumToParts	Nein	Nein	Ja (5.0.1+)	Nein	Nein
$datumToString	Ja	Ja	Ja	Ja	Ja
$DateTrunc	-	-	Nein	Ja	Nein
$TagOfMonth	Ja	Ja	Ja	Ja	Ja
$Tag OfWeek	Ja	Ja	Ja	Ja	Ja
$Tag OfYear	Ja	Ja	Ja	Ja	Ja
$hour	Ja	Ja	Ja	Ja	Ja
$ isoDayOfWeek	Ja	Ja	Ja	Ja	Ja
$isoWeek	Ja	Ja	Ja	Ja	Ja
$ isoWeekYear	Ja	Ja	Ja	Ja	Ja
$millisecond	Ja	Ja	Ja	Ja	Ja
$minute	Ja	Ja	Ja	Ja	Ja
$month	Ja	Ja	Ja	Ja	Ja
$second	Ja	Ja	Ja	Ja	Ja

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$week	Ja	Ja	Ja	Ja	Ja
$year	Ja	Ja	Ja	Ja	Ja

Wörtlicher Operator

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$literal	Ja	Ja	Ja	Ja	Ja

Operator zusammenführen

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$mergeObjects	Ja	Ja	Ja	Ja	Ja

Natürlicher Operator

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$ natürlich	Ja	Ja	Ja	Ja	Ja

Satzoperatoren

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$ allesElementsTrue	Nein	Ja	Ja	Ja	Ja
$ beliebigElementTrue	Nein	Ja	Ja	Ja	Ja
$setDifference	Nein	Ja	Ja	Ja	Ja
$setEquals	Ja	Ja	Ja	Ja	Ja
$setIntersection	Ja	Ja	Ja	Ja	Ja
$ gesetztIsSubset	Ja	Ja	Ja	Ja	Ja
$setUnion	Ja	Ja	Ja	Ja	Ja

Operatoren auf der Bühne

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$addFields	Ja	Ja	Ja	Ja	Ja
$bucket	Nein	Nein	Nein	Ja	Nein
$bucketAuto	Nein	Nein	Nein	Ja (8.0.2+)	Nein
$ Streamändern	Ja	Ja	Ja	Ja	Nein

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$ ändern StreamSplitLargeEvent	Nein	Nein	Nein	Ja (8.0.2+)	Nein
$collStats	Nein	Ja	Ja	Ja	Nein
$count	Ja	Ja	Ja	Ja	Ja
$currentOp	Ja	Ja	Ja	Ja	Ja
$ verdichten	Nein	Nein	Nein	Nein	Nein
$facet	Nein	Nein	Nein	Ja (8.0.2+)	Nein
$ füllen	Nein	Nein	Nein	Nein	Nein
$geoNear	Ja	Ja	Ja	Ja	Ja
$graphLookup	Nein	Nein	Nein	Ja	Nein
$group	Ja	Ja	Ja	Ja	Ja
$indexStats	Ja	Ja	Ja	Ja	Nein
$limit	Ja	Ja	Ja	Ja	Ja
$ liste LocalSessions	Nein	Nein	Nein	Nein	Nein
$listSessions	Nein	Nein	Nein	Nein	Nein
$lookup	Ja	Ja	Ja	Ja	Ja
$match	Ja	Ja	Ja	Ja	Ja

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$ zusammenführen	-	-	Nein	Ja	Nein
$out	Ja	Ja	Ja	Ja	Nein
$planen CacheStats	-	-	Nein	Nein	Nein
$project	Ja	Ja	Ja	Ja	Ja
$redact	Ja	Ja	Ja	Ja	Ja
$replaceRoot	Ja	Ja	Ja	Ja	Ja
$sample	Ja	Ja	Ja	Ja	Ja
$set	-	-	Nein	Ja	Nein
$ setzen WindowFields	-	-	Nein	Ja (8.0.2+)	Nein
$skip	Ja	Ja	Ja	Ja	Ja
$sort	Ja	Ja	Ja	Ja	Ja
$ sortieren ByCount	Nein	Nein	Nein	Ja (8.0.1+)	Nein
$ Union mit	-	-	Nein	Nein	Nein
$unset	-	-	Nein	Ja	Nein
$unwind	Ja	Ja	Ja	Ja	Ja
$ ersetzend urch	Nein	Nein	Nein	Ja	Nein

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$VectorSearch	Nein	Nein	Nein	Ja	Nein
$ listeSearchIndexes	-	-	-	Ja (8.0.1+)	Nein

Zeichenfolgenoperatoren

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$concat	Ja	Ja	Ja	Ja	Ja
$ indexOfBytes	Ja	Ja	Ja	Ja	Ja
$indexOfCP	Ja	Ja	Ja	Ja	Ja
$ ltrim	Nein	Ja	Ja	Ja	Nein
$RegexFind	-	-	Ja	Ja	Nein
$ regulärer AusdruckFindAll	-	-	Ja	Ja	Nein
$RegexMatch	-	-	Ja	Ja	Nein
$ Allesersetzen	-	-	Ja	Ja	Nein
$ ersetzenEins	-	-	Ja	Ja	Nein

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$rtrim	Nein	Ja	Ja	Ja	Nein
$split	Ja	Ja	Ja	Ja	Ja
$strcasecmp	Ja	Ja	Ja	Ja	Ja
$str LenBytes	Ja	Ja	Ja	Ja	Ja
$strLenCP	Ja	Ja	Ja	Ja	Ja
$substr	Ja	Ja	Ja	Ja	Ja
$substrBytes	Ja	Ja	Ja	Ja	Ja
$substrCP	Ja	Ja	Ja	Ja	Ja
$toLower	Ja	Ja	Ja	Ja	Ja
$toUpper	Ja	Ja	Ja	Ja	Ja
$trim	Nein	Ja	Ja	Ja	Nein

Systemvariablen

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$$CURRENT	Nein	Nein	Nein	Nein	Nein
$$DESCEND	Ja	Ja	Ja	Ja	Ja
$$KEEP	Ja	Ja	Ja	Ja	Ja
$$PRUNE	Ja	Ja	Ja	Ja	Ja
$$REMOVE	Nein	Nein	Nein	Nein	Nein

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$ WURZEL	Ja	Ja	Ja	Ja	Ja

Operator für die Textsuche

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$meta	Nein	Nein	Ja	Ja	Nein
$suche	Nein	Nein	Ja	Ja	Nein

Operatoren für die Typkonvertierung

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$ konvertieren	Nein	Ja	Ja	Ja	Ja
$ ist Zahl	-	-	Nein	Ja (8.0.1+)	Nein
$ zu Bool	Nein	Ja	Ja	Ja	Ja
$ bis Datum	Nein	Ja	Ja	Ja	Ja
$ zu Dezimal	Nein	Ja	Ja	Ja	Ja
$ bis verdoppeln	Nein	Ja	Ja	Ja	Ja
$ zu Int	Nein	Ja	Ja	Ja	Ja
$ zu Long	Nein	Ja	Ja	Ja	Ja

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$ bis ObjectId	Nein	Ja	Ja	Ja	Ja
$ toString	Nein	Ja	Ja	Ja	Ja
$ zu UUID	-	-	-	Ja (8.0.1+)	Nein

Variable Operatoren

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$let	Ja	Ja	Ja	Ja	Ja
$map	Ja	Ja	Ja	Ja	Ja

Trigonometrie-Operatoren

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$ acos	-	-	-	Ja (8.0.1+)	Nein
$ eine Münze	-	-	-	Ja (8.0.1+)	Nein
$ ASIN	-	-	-	Ja (8.0.1+)	Nein
$asin	-	-	-	Ja (8.0.1+)	Nein
$atan	-	-	-	Ja (8.0.1+)	Nein
$atan2	-	-	-	Ja (8.0.1+)	Nein
$atanh	-	-	-	Ja (8.0.1+)	Nein
$cos	-	-	-	Ja (8.0.1+)	Nein

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$cosh	-	-	-	Ja (8.0.1+)	Nein
$ Grad ToRadians	-	-	-	Ja (8.0.1+)	Nein
$ Radiant ToDegrees	-	-	-	Ja (8.0.1+)	Nein
$sin	-	-	-	Ja (8.0.1+)	Nein
$sinh	-	-	-	Ja (8.0.1+)	Nein
$tan	-	-	-	Ja (8.0.1+)	Nein
$tanh	-	-	-	Ja (8.0.1+)	Nein

Bitweise Operatoren (Aggregation)

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$bit und	-	-	-	Ja (8.0.1+)	Nein
$ Bit nicht	-	-	-	Ja (8.0.1+)	Nein
$ Bit oder	-	-	-	Ja (8.0.1+)	Nein
$ bitX oder	-	-	-	Ja (8.0.1+)	Nein

Zeitstempel-Operatoren

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$ts Inkrement	-	-	-	Ja (8.0.1+)	Nein
$ ts Sekunde	-	-	-	Ja (8.0.1+)	Nein

Verschiedene Betreiber

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
$ getField	-	-	Ja	Ja	Nein
$ Rand	-	-	Ja (5.0.1+)	Ja	Nein
$ Beispielrate	-	-	Nein	Ja (8.0.1+)	Nein
$ sigmoid	-	-	-	Ja (8.0.1+)	Nein

Datentypen

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
32-Bit-Ganzzahl (int)	Ja	Ja	Ja	Ja	Ja
64-Bit-Ganzzahl (lang)	Ja	Ja	Ja	Ja	Ja
Array	Ja	Ja	Ja	Ja	Ja
Binäre Daten	Ja	Ja	Ja	Ja	Ja

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
Boolesch	Ja	Ja	Ja	Ja	Ja
Date	Ja	Ja	Ja	Ja	Ja
DBPointer	Nein	Nein	Nein	Nein	Nein
dBrefs	Nein	Nein	Nein	Nein	Nein
Decimal128	Ja	Ja	Ja	Ja	Ja
Double	Ja	Ja	Ja	Ja	Ja
JavaScript	Nein	Nein	Nein	Nein	Nein
JavaScript(mit Umfang)	Nein	Nein	Nein	Nein	Nein
MaxKey	Ja	Ja	Ja	Ja	Ja
MinKey	Ja	Ja	Ja	Ja	Ja
Null	Ja	Ja	Ja	Ja	Ja
Objekt	Ja	Ja	Ja	Ja	Ja
ObjectId	Ja	Ja	Ja	Ja	Ja
Regulärer Ausdruck	Ja	Ja	Ja	Ja	Ja
Zeichenfolge	Ja	Ja	Ja	Ja	Ja
Symbol	Nein	Nein	Nein	Nein	Nein
Zeitstempel	Ja	Ja	Ja	Ja	Ja
Undefined	Nein	Nein	Nein	Nein	Nein

Indizes und Indexeigenschaften

Themen

- Indizes
- Eigenschaften indizieren

Indizes

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
2dsphere	Ja	Ja	Ja	Ja	Ja
2D-Index	Nein	Nein	Nein	Nein	Nein
Verbundindex	Ja	Ja	Ja	Ja	Ja
Hash-Index	Nein	Nein	Nein	Nein	Nein
Multikey-Index	Ja	Ja	Ja	Ja	Ja
Einzelfeldindex	Ja	Ja	Ja	Ja	Ja
Textindex	Nein	Nein	Ja	Ja	Nein
Platzhalter	Nein	Nein	Nein	Nein	Nein

Eigenschaften indizieren

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
Hintergrund	Ja	Ja	Ja	Ja	Ja

Befehl	3.6	4,0	5.0	8.0	Elastischer Cluster
Berücksichtigt Groß- und Kleinschreibung nicht	Nein	Nein	Nein	Ja	Nein
Hidden	Nein	Nein	Nein	Ja (8.0.1+)	Nein
Teilweise	Nein	Nein	Ja	Ja	Nein
Sparse	Ja	Ja	Ja	Ja	Ja
Text	Nein	Nein	Ja	Ja	Nein
TTL	Ja	Ja	Ja	Ja	Ja
Eindeutig	Ja	Ja	Ja	Ja	Ja
Vektor	Nein	Nein	Ja	Ja	Nein

Detaillierte Informationen zu bestimmten MongoDB-Operatoren finden Sie in den folgenden Themen:

- Betreiber von Aggregationspipelines
- Geodaten
- Operatoren für Projektionen
- Operatoren aktualisieren

Betreiber von Aggregationspipelines

Dieser Abschnitt enthält detaillierte Informationen zu Aggregationspipeline-Betreibern, die von Amazon DocumentDB unterstützt werden.

Themen

- $$DESCEND
- $$KEEP

- $$PRUNE
- $ WURZEL
- $abs
- $ acos
- $ eine Münze
- $add
- $hinzufügen ToSet
- $addFields
- $ alles ElementsTrue
- $and
- $ beliebig ElementTrue
- $array ElemAt
- $array ToObject
- $ ASIN
- $asin
- $atan
- $atan2
- $atanh
- $avg
- $ Binärgröße
- $bit und
- $ Bit nicht
- $ Bit oder
- $ bitX oder
- $ unten
- $ unten N
- $ bsonGröße
- $bucket
- $bucketAuto

- $ceil
- $ Stream ändern
- $ ändern StreamSplitLargeEvent
- $cmp
- $collStats
- $concat
- $concatArrays
- $cond
- $ konvertieren
- $cos
- $cosh
- $count (Stufe)
- $count (Akkumulator)
- $ CovariancePop
- $KovarianzSAMP
- $currentOp
- $DateAdd
- $DateDiff
- $datum FromParts
- $datum FromString
- $DateSubtrahieren
- $datum ToParts
- $datum ToString
- $DateTrunc
- $Tag OfMonth
- $Tag OfWeek
- $Tag OfYear
- $ Grad ToRadians
- $ Dichter Rang

- $divide
- $ Dokumentnummer
- $eq
- $exp
- $facet
- $filter
- $first
- $FirstN
- $floor
- $geoNear
- $graphLookup
- $group
- $gt
- $gte
- $hour
- $ifNull
- $in
- $ index OfArray
- $ index OfBytes
- $indexOfCP
- $indexStats
- $isArray
- $ ist Zahl
- $ iso DayOfWeek
- $isoWeek
- $ iso WeekYear
- $last
- $ letztes N
- $let
- $limit

- $ liste SearchIndexes
- $literal
- $ln
- $log
- $log10
- $lt
- $lte
- $lookup
- $ ltrim
- $map
- $match
- $max
- $ MaxN
- $ Median
- $meta
- $ zusammenführen
- $mergeObjects
- $millisecond
- $min
- $ min
- $minute
- $mod
- $month
- $multiply
- $ natürlich
- $ne
- $not
- $objekt ToArray
- $or
- $out

- $ Perzentil
- $pow
- $push
- $project
- $ Radiant ToDegrees
- $ Rand
- $range
- $ Rang
- $redact
- $reduce
- $RegexFind
- $ regulärer Ausdruck FindAll
- $RegexMatch
- $ Alles ersetzen
- $ ersetzenEins
- $replaceRoot
- $ ersetzendurch
- $reverseArray
- $ Runde
- $rtrim
- $sample
- $suche
- $second
- $set
- $setDifference
- $setEquals
- $setIntersection
- $ gesetzt IsSubset
- $setUnion
- $ setzen WindowFields

- $ sigmoid
- $sin
- $sinh
- $skip
- $slice
- $size
- $sort
- $sortArray
- $ sortieren ByCount
- $split
- $sqrt
- $ std DevPop
- $ std DevSamp
- $str LenBytes
- $strLenCP
- $strcasecmp
- $substr
- $substrBytes
- $substrCP
- $subtract
- $sum
- $switch
- $tan
- $tanh
- $top
- $ zu PN
- $trunc
- $ts Inkrement
- $ ts Sekunde
- $ zu Bool

- $ bis Datum
- $ zu Dezimal
- $ bis verdoppeln
- $ zu Int
- $ zu Long
- $toLower
- $ bis ObjectId
- $ toString
- $toUpper
- $ zu UUID
- $trim
- $type
- $unset
- $unwind
- $VectorSearch
- $week
- $year
- $zip

$$DESCEND

Der `$$DESCEND` Operator in Amazon DocumentDB ist ein spezieller Positions-Array-Operator, der in der `$redact` Pipeline-Phase verwendet wird. Er weist die Aggregationspipeline an, in das aktuelle Dokument abzusteigen und alle Felder zu verarbeiten, unabhängig von ihrer Verschachtelungsebene.

Wenn die `$redact` Phase auf den `$$DESCEND` Operator trifft, bleiben alle Felder im aktuellen Dokument sichtbar und werden weiter unten in der Pipeline verarbeitet. Dies ist nützlich, wenn Sie bestimmte Felder anhand einer Bedingung selektiv schwärzen oder beschneiden und dabei die Struktur des Dokuments beibehalten möchten.

Parameter

Keine.

Beispiel (MongoDB Shell)

In diesem Beispiel verwenden wir den `$redact` Schritt mit dem `$$DESCEND` Operator, um selektiv Dokumente anzuzeigen, bei denen das `code` Feld „Reg“ entspricht.

Erstellen Sie Beispieldokumente

```
db.patient.insertMany([
  { "_id": 1, "code": "Emp", "patient": "John Doe", "DOB": "1/1/1980", "Hospital":
 "Main" },
  { "_id": 2, "code": "Reg", "patient": "Jane Doe", "DOB": "3/27/1989", "Hospital":
 "Cherry Hill" },
  { "_id": 3, "code": "Emp", "patient": "Bob Smith", "DOB": "6/15/1975", "Hospital":
 "Downtown" }
]);
```

Beispiel für eine Abfrage

```
db.patient.aggregate([
  { $redact: {
    $cond: {
      if: { $eq: ["Reg", "$code"] },
      then: "$$DESCEND",
      else: "$$PRUNE"
    }
  }}
]);
```

Ausgabe

```
{
  "_id": 2,
  "code": "Reg",
  "patient": "Jane Doe",
  "DOB": "3/27/1989",
  "Hospital": "Cherry Hill"
}
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$$DESCEND` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function redactPatients() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('patient');

  const result = await collection.aggregate([
    { $redact: {
      $cond: {
        if: { $eq: ["Reg", "$code"] },
        then: "$$DESCEND",
        else: "$$PRUNE"
      }
    }}
  ]).toArray();

  console.log(result);
  client.close();
}

redactPatients();
```

Python

```
from pymongo import MongoClient

def redact_patients():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.patient

    result = list(collection.aggregate([
        { "$redact": {
            "$cond": {
                "if": { "$eq": ["Reg", "$code"] },
                "then": "$$DESCEND",
```

```
                "else": "$$PRUNE"
            }
        }}
    ]))

    print(result)
    client.close()

redact_patients()
```

$$KEEP

Die $$KEEP Systemvariable wird zusammen mit der $redact Phase in der Aggregationspipeline verwendet, um das aktuelle Dokument oder Feld unverändert zu lassen und es in die Ausgabe aufzunehmen.

Parameter

Keine

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung von $$KEEP in einer Amazon DocumentDB-Aggregationspipeline. Dokumente werden nur aufbewahrt, wenn der Zugriff auf „öffentlich“ steht, andernfalls werden sie entfernt.

Erstellen Sie Beispieldokumente

```
db.articles.insertMany([
  { title: "Article A", access: "public", content: "Visible content" },
  { title: "Article B", access: "private", content: "Hidden content" }
]);
```

Beispiel für eine Abfrage

```
db.articles.aggregate([
  {
    $redact: {
      $cond: [
        { $eq: ["$access", "public"] },
        "$$KEEP",
```

```
        "$$PRUNE"
      ]
    }
  }
]);
```

Ausgabe

```
[
  {
    "_id" : ObjectId("..."),
    "title" : "Article A",
    "access" : "public",
    "content" : "Visible content"
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des $$KEEP Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function run() {
  const client = new MongoClient(
    'mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-bundle.pem&replicaSet=rs0'
  );

  try {
    await client.connect();
    const db = client.db('test');
    const articles = db.collection('articles');

    const pipeline = [
      {
        $redact: {
          $cond: [
            { $eq: ["$access", "public"] },
            "$$KEEP",
```

```
            "$$PRUNE"
          ]
        }
      }
    ];

    const results = await articles.aggregate(pipeline).toArray();
    console.log(results);
  } finally {
    await client.close();
  }
}

run().catch(console.error);
```

Python

```
from pymongo import MongoClient

client = MongoClient(
    "mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-bundle.pem&replicaSet=rs0"
)

db = client.test
articles = db.articles

pipeline = [
    {
        "$redact": {
            "$cond": [
                {"$eq": ["$access", "public"]},
                "$$KEEP",
                "$$PRUNE"
            ]
        }
    }
]

results = list(articles.aggregate(pipeline))
print(results)

client.close()
```

$$PRUNE

Die $$PRUNE Systemvariable wird zusammen mit der $redact Phase in der Aggregationspipeline verwendet, um Dokumente oder eingebettete Dokumentebenen aus den Ergebnissen auszuschließen. Wenn eine Bedingung als Ergebnis ausgewertet wird$$PRUNE, wird das aktuelle Dokument oder Unterdokument aus der Ausgabe entfernt. Es wird normalerweise zusammen mit $$DESCEND (um das Dokument zu behalten und zu durchsuchen) oder $$KEEP (um das Dokument auf allen Ebenen beizubehalten) verwendet.

Parameter

Keine. Die $$PRUNE Systemvariable wird ohne Parameter verwendet und muss mit $redact verwendet werden.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie Sie $$PRUNE mit with Benutzer$redact, die älter als 30 Jahre sind, von den Ergebnissen ausschließen können.

Erstellen Sie Beispieldokumente

```
db.users.insert([
  { _id:1, name: "Carlos Salazar", age: 35, address: { street: "123 Main St", city:
 "Anytown", state: "CA" } },
  { _id:2, name: "Saanvi Sarkar", age: 28, address: { street: "456 Oak Rd", city:
 "Someplace", state: "NY" } },
  { _id:3, name: "Li Juan", age: 42, address: { street: "789 Pine Ave", city:
 "Springfield", state: "TX" } }
])
```

Beispiel für eine Abfrage

```
db.users.aggregate([
  {
    $redact: {
      $cond: {
        if: { $gt: ["$age", 30] },
        then: "$$PRUNE",
        else: "$$DESCEND"
      }
    }
  }
```

```
])
```

Ausgabe

```
[
  {
    "_id": 2,
    "name": "Saanvi Sarkar",
    "age": 28,
    "address": {
      "street": "456 Oak Rd",
      "city": "Someplace",
      "state": "NY"
    }
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des $$PRUNE Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

async function main() {
  await client.connect();
  const db = client.db('test');
  const users = db.collection('users');

  const result = await users.aggregate([
    {
      $redact: {
        $cond: {
          if: { $gt: ["$age", 30] },
          then: "$$PRUNE",
          else: "$$DESCEND"
        }
```

```
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
db = client['test']
users = db['users']

result = list(users.aggregate([
    {
        '$redact': {
            '$cond': {
                'if': { '$gt': ['$age', 30] },
                'then': '$$PRUNE',
                'else': '$$DESCEND'
            }
        }
    }
]))

print(result)
client.close()
```

$ WURZEL

Der $R00T Operator in Amazon DocumentDB wird verwendet, um auf das gesamte Eingabedokument innerhalb einer Aggregationspipeline zu verweisen. Damit können Sie auf das gesamte Dokument zugreifen und es bearbeiten, einschließlich all seiner verschachtelten Felder und Strukturen.

Parameter

Keine

Beispiel (MongoDB Shell)

Dieses Beispiel zeigt, wie $ROOT Sie ein Prüfprotokoll erstellen können, das das vollständige Originaldokument zusammen mit Metadaten darüber, wann es verarbeitet wurde, erfasst.

Erstellen Sie Beispieldokumente

```
db.orders.insertMany([
  {
    _id: 1,
    orderId: "ORD-2024-001",
    customer: "María García",
    email: "maría@example.com",
    items: [
      { product: "Laptop", quantity: 1, price: 1299.99 }
    ],
    totalAmount: 1299.99
  },
  {
    _id: 2,
    orderId: "ORD-2024-002",
    customer: "Arnav Desai",
    email: "arnav@example.com",
    items: [
      { product: "Mouse", quantity: 2, price: 29.99 },
      { product: "Keyboard", quantity: 1, price: 89.99 }
    ],
    totalAmount: 149.97
  }
]);
```

Beispiel für eine Abfrage

```
db.orders.aggregate([
  {
    $project: {
      processedAt: new Date(),
      originalDocument: "$$ROOT",
      summary: {
```

```
        $concat: [
          "Order ",
          "$orderId",
          " for ",
          "$customer",
          " - Total: $",
          { $toString: "$totalAmount" }
        ]
      }
    }
  }
]);
```

Ausgabe

```
[
  {
    _id: 1,
    processedAt: ISODate('2025-11-24T20:43:51.492Z'),
    originalDocument: {
      _id: 1,
      orderId: 'ORD-2024-001',
      customer: 'María García',
      email: 'maría@example.com',
      items: [ { product: 'Laptop', quantity: 1, price: 1299.99 } ],
      totalAmount: 1299.99
    },
    summary: 'Order ORD-2024-001 for María García - Total: $1299.99'
  },
  {
    _id: 2,
    processedAt: ISODate('2025-11-24T20:43:51.492Z'),
    originalDocument: {
      _id: 2,
      orderId: 'ORD-2024-002',
      customer: 'Arnav Desai',
      email: 'arnav@example.com',
      items: [
        { product: 'Mouse', quantity: 2, price: 29.99 },
        { product: 'Keyboard', quantity: 1, price: 89.99 }
      ],
      totalAmount: 149.97
    },
```

```
    summary: 'Order ORD-2024-002 for Arnav Desai - Total: $149.97'
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des $ROOT Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function createAuditLog() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const orders = db.collection('orders');

  const result = await orders.aggregate([
    {
      $project: {
        processedAt: new Date(),
        originalDocument: "$$ROOT",
        summary: {
          $concat: [
            "Order ",
            "$orderId",
            " for ",
            "$customer",
            " - Total: $",
            { $toString: "$totalAmount" }
          ]
        }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}
```

```
createAuditLog();
```

Python

```
from pymongo import MongoClient
from datetime import datetime

def create_audit_log():
    client = MongoClient('mongodb://username:password@docdb-
cluster.cluster-123456789.us-east-1.docdb.amazonaws.com:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['orders']

    result = list(collection.aggregate([
        {
            '$project': {
                'processedAt': datetime.now(),
                'originalDocument': '$$ROOT',
                'summary': {
                    '$concat': [
                        "Order ",
                        "$orderId",
                        " for ",
                        "$customer",
                        " - Total: $",
                        { '$toString': "$totalAmount" }
                    ]
                }
            }
        }
    ]))

    print(result)
    client.close()

create_audit_log()
```

$abs

Der `$abs` Operator in Amazon DocumentDB gibt den absoluten Wert einer Zahl zurück. Er kann in der Aggregationspipeline verwendet werden, um mathematische Operationen an numerischen Feldern durchzuführen.

Parameter

- `number`: Der numerische Ausdruck, für den der absolute Wert zurückgegeben wird.

Beispiel (MongoDB Shell)

Dieses Beispiel demonstriert die Verwendung des `$abs` Operators, um den absoluten Wert eines numerischen Feldes zu ermitteln.

Erstellen Sie Beispieldokumente

```
db.numbers.insertMany([
  { "_id": 1, "value": -5 },
  { "_id": 2, "value": 10 },
  { "_id": 3, "value": -3.14 },
  { "_id": 4, "value": 0 }
]);
```

Beispiel für eine Abfrage

```
db.numbers.aggregate([
  { $project: {
    "_id": 1,
    "absolute_value": { $abs: "$value" }
  }}
]);
```

Ausgabe

```
[
  { "_id": 1, "absolute_value": 5 },
  { "_id": 2, "absolute_value": 10 },
  { "_id": 3, "absolute_value": 3.14 },
  { "_id": 4, "absolute_value": 0 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$abs` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('numbers');

  const result = await collection.aggregate([
    { $project: {
      "_id": 1,
      "absolute_value": { $abs: "$value" }
    }}
  ]).toArray();

  console.log(result);
  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['numbers']

    result = list(collection.aggregate([
        { '$project': {
            "_id": 1,
```

```
            "absolute_value": { "$abs": "$value" }
        }}
    ]))

    print(result)
    client.close()

if __name__ == "__main__":
    main()
```

$ acos

Neu ab Version 8.0.1.

Der `$acos` Operator in Amazon DocumentDB gibt den Arkuskosinus (umgekehrten Kosinus) eines Werts zurück. Der Eingabewert muss zwischen -1 und 1 liegen.

Parameter

- `expression`: Ein Ausdruck, der in eine Zahl zwischen -1 und 1 aufgelöst wird.

Das Ergebnis ist im Bogenmaß angegeben. Um Grade zu erhalten, wenden Sie `$radiansToDegrees` sich an die Ausgabe.

Der Rückgabetyp ist `double` standardmäßig. Wenn die Eingabe eine 128-Bit-Dezimalzahl ist, ist die Ausgabe ebenfalls eine 128-Bit-Dezimalzahl.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$acos` Operator verwendet wird, um den Arkuskosinus von numerischen Werten zu berechnen.

Erstellen Sie Beispieldokumente

```
db.values.insertMany([
  { "_id": 1, "value": 1 },
  { "_id": 2, "value": 0 },
  { "_id": 3, "value": -1 }
]);
```

Beispiel für eine Abfrage

```
db.values.aggregate([
  { $project: {
    "angle": { $acos: "$value" }
  }}
]);
```

Ausgabe

```
[
  { "_id": 1, "angle": 0 },
  { "_id": 2, "angle": 1.5707963267948966 },
  { "_id": 3, "angle": 3.141592653589793 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$acos` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('values');

  const result = await collection.aggregate([
    { $project: {
      "angle": { $acos: "$value" }
    }}
  ]).toArray();

  console.log(result);
  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['values']

    result = list(collection.aggregate([
        { '$project': {
            'angle': { '$acos': '$value' }
        }}
    ]))

    print(result)
    client.close()

if __name__ == "__main__":
    main()
```

$ eine Münze

Neu ab Version 8.0.1.

Der `$acosh` Operator in Amazon DocumentDB gibt den inversen hyperbolischen Kosinus (hyperbolischer Arkuskosinus) eines Werts zurück. Der Eingabewert muss größer oder gleich 1 sein.

Parameter

- `expression`: Ein Ausdruck, der in eine Zahl größer oder gleich 1 aufgelöst wird.

Der Rückgabetyp ist `double` standardmäßig. Wenn die Eingabe eine 128-Bit-Dezimalzahl ist, ist die Ausgabe ebenfalls eine 128-Bit-Dezimalzahl.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$acosh` Operator verwendet wird, um den inversen hyperbolischen Kosinus numerischer Werte zu berechnen.

Erstellen Sie Beispieldokumente

```
db.values.insertMany([
  { "_id": 1, "value": 1 },
  { "_id": 2, "value": 2 },
  { "_id": 3, "value": 10 }
]);
```

Beispiel für eine Abfrage

```
db.values.aggregate([
  { $project: {
    "result": { $acosh: "$value" }
  }}
]);
```

Ausgabe

```
[
  { "_id": 1, "result": 0 },
  { "_id": 2, "result": 1.3169578969248166 },
  { "_id": 3, "result": 2.993222846126381 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$acosh` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('values');

  const result = await collection.aggregate([
```

```
    { $project: {
      "result": { $acosh: "$value" }
    }}
  ]).toArray();

  console.log(result);
  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['values']

    result = list(collection.aggregate([
        { '$project': {
            'result': { '$acosh': '$value' }
        }}
    ]))

    print(result)
    client.close()

if __name__ == "__main__":
    main()
```

$add

Der `$add` Operator in Amazon DocumentDB wird verwendet, um Zahlen oder Daten zu addieren. Er kann verwendet werden, um arithmetische Operationen an numerischen Feldern durchzuführen oder um Datumsarithmetik durchzuführen, indem einem Datumsfeld eine Anzahl von Zeiteinheiten hinzugefügt wird.

Parameter

- `expression1`: Die erste Zahl oder das erste Datum, das hinzugefügt werden soll.
- `expression2`: Die zweite Zahl oder das zweite Datum, das hinzugefügt werden soll.
- `expression3`: (optional) Zusätzliche Zahlen oder Daten, die hinzugefügt werden sollen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$add` Operator verwendet wird, um zwei Zahlen zu addieren.

Erstellen Sie Beispieldokumente

```
db.numbers.insertMany([
  { _id: 1, a: 5, b: 3 },
  { _id: 2, a: 10, b: 7 },
  { _id: 3, a: 2, b: 8 }
]);
```

Beispiel für eine Abfrage

```
db.numbers.aggregate([
  { $project: {
    _id: 1,
    sum: { $add: ["$a", "$b"] }
  }}
])
```

Ausgabe

```
[
  { "_id" : 1, "sum" : 8 },
  { "_id" : 2, "sum" : 17 },
  { "_id" : 3, "sum" : 10 }
]
```

In diesem Beispiel wird der `$add` Operator verwendet, um die Werte der b Felder a und für jedes Dokument hinzuzufügen und das Ergebnis im `sum` Feld zu speichern.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$add` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('numbers');

  const result = await collection.aggregate([
    { $project: {
      _id: 1,
      sum: { $add: ['$a', '$b'] }
    }}
  ]).toArray();

  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
db = client['test']
collection = db['numbers']

result = list(collection.aggregate([
    { '$project': {
        '_id': 1,
        'sum': { '$add': ['$a', '$b'] }
    }}
]))

print(result)
client.close()
```

$hinzufügen ToSet

Der `$addToSet` Aggregationsoperator gibt ein Array mit eindeutigen Werten aus einem angegebenen Ausdruck für jede Gruppe zurück. Er wird innerhalb der `$group` Phase verwendet, um unterschiedliche Werte zu akkumulieren und Duplikate automatisch zu eliminieren.

Parameter

- `expression`: Der Ausdruck, der für jedes Dokument in der Gruppe ausgewertet werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$addToSet` Operator verwendet wird, um für jeden Kunden eindeutige Städte zu erfassen, in denen Bestellungen aufgegeben wurden.

Erstellen Sie Beispieldokumente

```
db.orders.insertMany([
  { _id: 1, customer: "Alice", city: "Seattle", amount: 100 },
  { _id: 2, customer: "Alice", city: "Portland", amount: 150 },
  { _id: 3, customer: "Bob", city: "Seattle", amount: 200 },
  { _id: 4, customer: "Alice", city: "Seattle", amount: 75 },
  { _id: 5, customer: "Bob", city: "Boston", amount: 300 }
]);
```

Beispiel für eine Abfrage

```
db.orders.aggregate([
  {
    $group: {
      _id: "$customer",
      cities: { $addToSet: "$city" }
    }
  }
]);
```

Ausgabe

```
[
  { _id: 'Bob', cities: [ 'Seattle', 'Boston' ] },
  { _id: 'Alice', cities: [ 'Seattle', 'Portland' ] }
```

```
]
```

Beispiel für die Verwendung von Fensteroperatoren (MongoDB Shell)

Neu ab Version 8.0.2.

Der `$addToSet` Operator kann auch als Fensterbediener auf der `$setWindowFields` Bühne verwendet werden. In diesem Zusammenhang gibt es ein Array mit den eindeutigen Werten des angegebenen Ausdrucks für die Dokumente in jedem Fenster zurück. Sie geben den Operator unter dem `output` Feld an und definieren optional die Fenstergrenzen mit einem `window` Dokument.

> **Note**
>
> Bei Verwendung als Fensteroperator in `$setWindowFields` `$addToSet` ist er auf 100 MB an Zwischendaten begrenzt. Eine Operation, die dieses Limit überschreitet, gibt einen Fehler zurück.

Erstellen Sie Beispieldokumente

```
db.visits.insertMany([
  { _id: 1, customer: "Alice", day: 1, city: "Seattle" },
  { _id: 2, customer: "Alice", day: 2, city: "Portland" },
  { _id: 3, customer: "Alice", day: 3, city: "Seattle" },
  { _id: 4, customer: "Bob", day: 1, city: "Boston" },
  { _id: 5, customer: "Bob", day: 2, city: "Boston" }
]);
```

Beispiel für eine Abfrage

Das folgende Beispiel partitioniert die Dokumente nach`customer`, sortiert jede Partition nach `day` und gibt die Gruppe der `city` Einzelwerte vom Anfang der Partition bis zum aktuellen Dokument zurück.

```
db.visits.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$customer",
      sortBy: { day: 1 },
      output: {
        citiesSoFar: {
```

```
          $addToSet: "$city",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "customer": "Alice", "day": 1, "city": "Seattle", "citiesSoFar":
 [ "Seattle" ] },
  { "_id": 2, "customer": "Alice", "day": 2, "city": "Portland", "citiesSoFar":
 [ "Portland", "Seattle" ] },
  { "_id": 3, "customer": "Alice", "day": 3, "city": "Seattle", "citiesSoFar":
 [ "Portland", "Seattle" ] },
  { "_id": 4, "customer": "Bob", "day": 1, "city": "Boston", "citiesSoFar":
 [ "Boston" ] },
  { "_id": 5, "customer": "Bob", "day": 2, "city": "Boston", "citiesSoFar":
 [ "Boston" ] }
]
```

Jedes Dokument wird vom Anfang bis zum aktuellen Dokument um die Gruppe von `city` Einzelwerten innerhalb seiner Partition erweitert. `citiesSoFar` Wie bei der Akkumulatorform ist die Reihenfolge der Werte im zurückgegebenen Array nicht garantiert.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$addToSet` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten. Die folgenden Beispiele zeigen sowohl die Verwendung des Akkumulators (in`$group`) als auch die Verwendung von Fensteroperatoren (in`$setWindowFields`):

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
```

```
const db = client.db('test');

// Accumulator usage: collect unique values per group
const orders = db.collection('orders');
await orders.insertMany([
  { _id: 1, customer: "Alice", city: "Seattle", amount: 100 },
  { _id: 2, customer: "Alice", city: "Portland", amount: 150 },
  { _id: 3, customer: "Bob", city: "Seattle", amount: 200 },
  { _id: 4, customer: "Alice", city: "Seattle", amount: 75 },
  { _id: 5, customer: "Bob", city: "Boston", amount: 300 }
]);
const accumulatorResult = await orders.aggregate([
  {
    $group: {
      _id: "$customer",
      cities: { $addToSet: "$city" }
    }
  }
]).toArray();
console.log('Accumulator result:', accumulatorResult);

// Window operator usage: running set of unique values within each partition
const visits = db.collection('visits');
await visits.insertMany([
  { _id: 1, customer: "Alice", day: 1, city: "Seattle" },
  { _id: 2, customer: "Alice", day: 2, city: "Portland" },
  { _id: 3, customer: "Alice", day: 3, city: "Seattle" },
  { _id: 4, customer: "Bob", day: 1, city: "Boston" },
  { _id: 5, customer: "Bob", day: 2, city: "Boston" }
]);
const windowResult = await visits.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$customer",
      sortBy: { day: 1 },
      output: {
        citiesSoFar: {
          $addToSet: "$city",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]).toArray();
```

```
  console.log('Window result:', windowResult);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']

    # Accumulator usage: collect unique values per group
    orders = db['orders']
    orders.insert_many([
        { '_id': 1, 'customer': 'Alice', 'city': 'Seattle', 'amount': 100 },
        { '_id': 2, 'customer': 'Alice', 'city': 'Portland', 'amount': 150 },
        { '_id': 3, 'customer': 'Bob', 'city': 'Seattle', 'amount': 200 },
        { '_id': 4, 'customer': 'Alice', 'city': 'Seattle', 'amount': 75 },
        { '_id': 5, 'customer': 'Bob', 'city': 'Boston', 'amount': 300 }
    ])
    accumulator_result = list(orders.aggregate([
        {
            '$group': {
                '_id': '$customer',
                'cities': { '$addToSet': '$city' }
            }
        }
    ]))
    print('Accumulator result:', accumulator_result)

    # Window operator usage: running set of unique values within each partition
    visits = db['visits']
    visits.insert_many([
        { '_id': 1, 'customer': 'Alice', 'day': 1, 'city': 'Seattle' },
        { '_id': 2, 'customer': 'Alice', 'day': 2, 'city': 'Portland' },
        { '_id': 3, 'customer': 'Alice', 'day': 3, 'city': 'Seattle' },
        { '_id': 4, 'customer': 'Bob', 'day': 1, 'city': 'Boston' },
```

```
        { '_id': 5, 'customer': 'Bob', 'day': 2, 'city': 'Boston' }
    ])
    window_result = list(visits.aggregate([
        {
            '$setWindowFields': {
                'partitionBy': '$customer',
                'sortBy': { 'day': 1 },
                'output': {
                    'citiesSoFar': {
                        '$addToSet': '$city',
                        'window': { 'documents': ['unbounded', 'current'] }
                    }
                }
            }
        }
    ]))
    print('Window result:', window_result)

    client.close()

example()
```

$addFields

Die `$addFields` Phase in der Amazon DocumentDB-Aggregationspipeline ermöglicht es Ihnen, neue berechnete Felder zu Dokumenten hinzuzufügen. Dies kann nützlich sein, um den Dokumenten abgeleitete oder transformierte Daten hinzuzufügen.

Parameter

- `newField`: Der Name des neuen Feldes, das hinzugefügt werden soll.
- `expression`: Ein Ausdruck, der den Wert des neuen Felds ergibt.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie Sie ein neues Feld `$addFields` hinzufügen`TotalInventory`, das den Gesamtbestand auf der Grundlage der Felder `Inventory.OnHand` und `Inventory.OrderQnty` berechnet.

Erstellen Sie Beispieldokumente

```
db.example.insertMany([
  {
    "Item": "Spray Paint",
    "Colors": ["Black", "Red", "Green", "Blue"],
    "Inventory": {
      "OnHand": 47,
      "MinOnHand": 50,
      "OrderQnty": 36
    },
    "UnitPrice": 3.99
  },
  {
    "Item": "Ruler",
    "Colors": ["Red", "Green", "Blue", "Clear", "Yellow"],
    "Inventory": {
      "OnHand": 47,
      "MinOnHand": 40
    },
    "UnitPrice": 0.89
  }
]);
```

Beispiel für eine Abfrage

```
db.example.aggregate([
  {
    $addFields: {
      TotalInventory: { $add: ["$Inventory.OnHand", "$Inventory.OrderQnty"] }
    }
  }
])
```

Ausgabe

```
[
  {
    "_id" : ObjectId("5bedafbcf65ff161707de24f"),
    "Item" : "Ruler",
    "Colors" : [ "Red", "Green", "Blue", "Clear", "Yellow" ],
    "Inventory" : {
      "OnHand" : 47,
      "MinOnHand" : 40
```

```
    },
    "UnitPrice" : 0.89,
    "TotalInventory" : 47
  },
  {
    "_id" : ObjectId("5bedafbcf65ff161707de250"),
    "Item" : "Spray Paint",
    "Colors" : [ "Black", "Red", "Green", "Blue" ],
    "Inventory" : {
      "OnHand" : 47,
      "MinOnHand" : 50,
      "OrderQnty" : 36
    },
    "UnitPrice" : 3.99,
    "TotalInventory" : 83
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$addFields` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
const db = client.db('test');
const collection = db.collection('example');

const result = await collection.aggregate([
  {
    $addFields: {
      TotalInventory: { $add: ['$Inventory.OnHand', '$Inventory.OrderQnty'] }
    }
  }
]).toArray();

console.log(result);
```

```
await client.close();
```

Python

```
from pymongo import MongoClient

client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
db = client['test']
collection = db['example']

result = list(collection.aggregate([
    {
        '$addFields': {
            'TotalInventory': { '$add': ['$Inventory.OnHand',
 '$Inventory.OrderQnty'] }
        }
    }
]))

print(result)
client.close()
```

$ alles ElementsTrue

Neu ab Version 4.0

Der `$allElementsTrue` Operator wird verwendet, um zu überprüfen, ob alle Elemente in einem Array einen wahren Wert ergeben.

Parameter

- `expression`: Ein Ausdruck, der zu einem Array ausgewertet wird.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung von`$allElementsTrue`, um zu überprüfen, ob alle Elemente in einem Array wahr sind.

Erstellen Sie Beispieldokumente

```
db.collection.insert([
  { "name": "John", "scores": [100, 90, 80] },
  { "name": "Jane", "scores": [80, 85, 0] },
  { "name": "Bob", "scores": [90, 95, null] }
])
```

Beispiel für eine Abfrage

```
db.collection.find({
  "scores": { "$allElementsTrue": [{ "$gt": 0 }] }
})
```

Ausgabe

```
[
  { "_id" : ObjectId("..."), "name" : "John", "scores" : [ 100, 90, 80 ] },
  { "_id" : ObjectId("..."), "name" : "Bob", "scores" : [ 90, 95, null ] }
]
```

In diesem Beispiel prüft die Abfrage, ob alle Elemente im `scores` Array größer als 0 sind. Das Dokument mit `"name": "Jane"` wird ausgeschlossen, weil das `scores` Array eine 0 enthält, was ein falscher Wert ist.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$allElementsTrue` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('collection');

  const result = await collection.find({
```

```
    "scores": { "$allElementsTrue": [{ "$gt": 0 }] }
  }).toArray();

  console.log(result);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['collection']

    result = list(collection.find({
        "scores": {"$allElementsTrue": [{"$gt": 0}]}
    }))

    print(result)

    client.close()

example()
```

$and

Der `$and` Aggregationsoperator wertet mehrere Ausdrücke aus und gibt `true` nur zurück, wenn alle Ausdrücke zu 0 ausgewertet werden. `true` Wenn ein Ausdruck „Ja“ ist`false`, wird er zurückgegeben. `false`

Parameter

- `expressions`: Eine Reihe von Ausdrücken, die ausgewertet werden sollen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der $and Operator verwendet wird, um zu überprüfen, ob Produkte mehrere Kriterien erfüllen.

Erstellen Sie Beispieldokumente

```
db.products.insertMany([
  { _id: 1, name: "Laptop", price: 1200, inStock: true },
  { _id: 2, name: "Mouse", price: 25, inStock: false },
  { _id: 3, name: "Keyboard", price: 75, inStock: true }
]);
```

Beispiel für eine Abfrage

```
db.products.aggregate([
  {
    $project: {
      name: 1,
      price: 1,
      inStock: 1,
      affordable: {
        $and: [
          { $lt: ["$price", 100] },
          { $eq: ["$inStock", true] }
        ]
      }
    }
  }
]);
```

Ausgabe

```
[
  { _id: 1, name: 'Laptop', price: 1200, inStock: true, affordable: false },
  { _id: 2, name: 'Mouse', price: 25, inStock: false, affordable: false },
  { _id: 3, name: 'Keyboard', price: 75, inStock: true, affordable: true }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des $and Aggregationsoperators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('products');

  const result = await collection.aggregate([
    {
      $project: {
        name: 1,
        price: 1,
        inStock: 1,
        affordable: {
          $and: [
            { $lt: ["$price", 100] },
            { $eq: ["$inStock", true] }
          ]
        }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
```

```
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['products']

    result = list(collection.aggregate([
        {
            '$project': {
                'name': 1,
                'price': 1,
                'inStock': 1,
                'affordable': {
                    '$and': [
                        { '$lt': ['$price', 100] },
                        { '$eq': ['$inStock', True] }
                    ]
                }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$ beliebig ElementTrue

Neu ab Version 4.0

Der `$anyElementTrue` Operator wird verwendet, um festzustellen, ob ein Element in einem Array wahr ist.

Parameter

- `field`: Ein auszuwertendes Array-Feld.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung von`$anyElementTrue`, um zu überprüfen, ob ein Element in einem Array wahr ist.

Erstellen Sie Beispieldokumente

```
db.grades.insertMany([
  { _id: 1, student: 'Tim', scores: [false, false, null] },
  { _id: 2, student: 'Bob', scores: [false, 0, false] },
  { _id: 3, student: 'Ivy', scores: [false, true, 0] }
])
```

Beispiel für eine Abfrage

```
db.grades.aggregate([
  {
    $project: {
      student: 1,
      isAnyTrue: { $anyElementTrue: ["$scores"] },
      _id: 0
    }
  }
])
```

Ausgabe

```
[
  { student: 'Tim', isAnyTrue: false },
  { student: 'Bob', isAnyTrue: false },
  { student: 'Ivy', isAnyTrue: true }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$anyElementTrue` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
```

```
  const db = client.db('test');
  const collection = db.collection('grades');

  const result = await collection.aggregate([
    {
      $project: {
        student: 1,
        isAnyTrue: { $anyElementTrue: ["$scores"] },
        _id: 0
      }
    }
  ]).toArray();

  console.log(result);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['grades']

    result = list(collection.aggregate([
      {
        "$project": {
          "student": 1,
          "isAnyTrue": { "$anyElementTrue": ["$scores"] },
          "_id": 0
        }
      }
    ]))

    print(result)
```

```
    client.close()

example()
```

$array ElemAt

Der `$arrayElemAt` Operator in Amazon DocumentDB ermöglicht es Ihnen, ein Element anhand seiner Indexposition aus einem Array abzurufen. Dies ist besonders nützlich, wenn Sie auf ein bestimmtes Element innerhalb eines Array-Felds in Ihren Dokumenten zugreifen müssen.

Parameter

- `array`: Das Eingabe-Array, aus dem ein Element abgerufen werden soll.
- `index`: Die auf Null basierende Indexposition des abzurufenden Elements. Dieser Wert muss eine nicht negative Ganzzahl sein.

Beispiel (MongoDB Shell)

In diesem Beispiel zeigen wir, wie der `$arrayElemAt` Operator verwendet wird, um bestimmte Elemente aus dem `flight_miles` Array in der `miles` Sammlung abzurufen.

Erstellen Sie Beispieldokumente

```
db.miles.insertMany([
  { "_id" : 1, "member_since" : ISODate("1987-01-01T00:00:00Z"), "credit_card" : false,
 "flight_miles" : [ 1205, 2560, 880 ]},
  { "_id" : 2, "member_since" : ISODate("1982-01-01T00:00:00Z"), "credit_card" : true,
 "flight_miles" : [ 1205, 2560, 890, 2780 ]},
  { "_id" : 3, "member_since" : ISODate("1999-01-01T00:00:00Z"), "credit_card" : true,
 "flight_miles" : [ 1205, 880 ]}
]);
```

Beispiel für eine Abfrage

```
db.miles.aggregate([
  { $project: {
    "_id": 1,
    "first_mile": { $arrayElemAt: [ "$flight_miles", 0 ] },
    "last_mile": { $arrayElemAt: [ "$flight_miles", -1 ] }
```

```
  }}
]);
```

Ausgabe

```
{ "_id" : 1, "first_mile" : 1205, "last_mile" : 880 }
{ "_id" : 2, "first_mile" : 1205, "last_mile" : 2780 }
{ "_id" : 3, "first_mile" : 1205, "last_mile" : 880 }
```

In diesem Beispiel verwenden wir den `$arrayElemAt` Operator, um das erste und das letzte Element des `flight_miles` Arrays für jedes Dokument abzurufen.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$arrayElemAt` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('miles');

  const result = await collection.aggregate([
    { $project: {
      "_id": 1,
      "first_mile": { $arrayElemAt: [ "$flight_miles", 0 ] },
      "last_mile": { $arrayElemAt: [ "$flight_miles", -1 ] }
    }}
  ]).toArray();

  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.mydatabase
    collection = db.miles

    result = list(collection.aggregate([
        { "$project": {
            "_id": 1,
            "first_mile": { "$arrayElemAt": [ "$flight_miles", 0 ] },
            "last_mile": { "$arrayElemAt": [ "$flight_miles", -1 ] }
        }}
    ]))

    print(result)
    client.close()

example()
```

$array ToObject

Der `$arrayToObject` Operator in Amazon DocumentDB ist das Gegenteil des `$objectToArray` Operators. Er nimmt ein Array von Dokumenten mit Schlüssel-Wert-Paaren und konvertiert es in ein einzelnes Dokument. Dies ist besonders nützlich, wenn Sie ein Array von Schlüssel-Wert-Paaren wieder in eine Objekt- oder Dokumentstruktur konvertieren müssen.

Parameter

- `array expression`: Ein Ausdruck, der in ein Array aufgelöst wird. Die Array-Elemente müssen Dokumente mit zwei Feldern sein: k (der Schlüssel) und v (der Wert).

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie Sie `$arrayToObject` ein Array von Schlüssel-Wert-Paaren wieder in ein Dokument konvertieren können.

Erstellen Sie Beispieldokumente

```
db.videos.insertMany([
  { "_id": 1, "name": "Live Soft", "inventory": { "Des Moines": 1000, "Ames": 500 } },
  { "_id": 2, "name": "Top Pilot", "inventory": { "Mason City": 250, "Des Moines":
 1000 } },
  { "_id": 3, "name": "Romancing the Rock", "inventory": { "Mason City": 250, "Ames":
 500 } },
  { "_id": 4, "name": "Bravemind", "inventory": { "Mason City": 250, "Des Moines":
 1000, "Ames": 500 } }
]);
```

Beispiel für eine Abfrage

```
db.videos.aggregate([
  { $project: { name: 1, videos: { $objectToArray: "$inventory" } } },
  { $project: { name: 1, inventory: { $arrayToObject: "$videos" } } }
]);
```

Ausgabe

```
{ "_id" : 1, "name" : "Live Soft", "inventory" : { "Des Moines" : 1000, "Ames" :
 500 } }
{ "_id" : 2, "name" : "Top Pilot", "inventory" : { "Mason City" : 250, "Des Moines" :
 1000 } }
{ "_id" : 3, "name" : "Romancing the Rock", "inventory" : { "Mason City" : 250,
 "Ames" : 500 } }
{ "_id" : 4, "name" : "Bravemind", "inventory" : { "Mason City" : 250, "Des Moines" :
 1000, "Ames" : 500 } }
```

In diesem Beispiel wird der `$objectToArray` Operator verwendet, um das `inventory` Objekt in ein Array von Schlüssel-Wert-Paaren umzuwandeln. Der `$arrayToObject` Operator wird dann verwendet, um das Array wieder in ein Dokument zu konvertieren, wodurch die ursprüngliche Objektstruktur wiederhergestellt wird.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$arrayToObject` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('videos');

  const result = await collection.aggregate([
    { $project: { name: 1, videos: { $objectToArray: '$inventory' } } },
    { $project: { name: 1, inventory: { $arrayToObject: '$videos' } } }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['videos']

    result = list(collection.aggregate([
        { '$project': { 'name': 1, 'videos': { '$objectToArray': '$inventory' } } },
        { '$project': { 'name': 1, 'inventory': { '$arrayToObject': '$videos' } } }
    ]))

    print(result)
    client.close()

example()
```

$ ASIN

Neu ab Version 8.0.1.

Der `$asin` Operator in Amazon DocumentDB gibt den Arkussinus (umgekehrter Sinus) eines Werts zurück. Der Eingabewert muss zwischen -1 und 1 liegen.

Parameter

- `expression`: Ein Ausdruck, der in eine Zahl zwischen -1 und 1 aufgelöst wird.

Das Ergebnis ist im Bogenmaß angegeben. Um Grade zu erhalten, wenden Sie `$radiansToDegrees` sich an die Ausgabe.

Der Rückgabetyp ist `double` standardmäßig. Wenn die Eingabe eine 128-Bit-Dezimalzahl ist, ist die Ausgabe ebenfalls eine 128-Bit-Dezimalzahl.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$asin` Operator verwendet wird, um den Arkussinus numerischer Werte zu berechnen.

Erstellen Sie Beispieldokumente

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 0.5 },
  { "_id": 3, "value": 1 }
]);
```

Beispiel für eine Abfrage

```
db.values.aggregate([
  { $project: {
    "angle": { $asin: "$value" }
  }}
]);
```

Ausgabe

```
[
```

```
  { "_id": 1, "angle": 0 },
  { "_id": 2, "angle": 0.5235987755982989 },
  { "_id": 3, "angle": 1.5707963267948966 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$asin` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('values');

  const result = await collection.aggregate([
    { $project: {
      "angle": { $asin: "$value" }
    }}
  ]).toArray();

  console.log(result);
  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['values']
```

```
    result = list(collection.aggregate([
        { '$project': {
            'angle': { '$asin': '$value' }
        }}
    ]))

    print(result)
    client.close()

if __name__ == "__main__":
    main()
```

$asin

Neu ab Version 8.0.1.

Der `$asinh` Operator in Amazon DocumentDB gibt den inversen hyperbolischen Sinus eines Werts zurück. Verwenden Sie ihn in der Aggregationspipeline, um hyperbolische trigonometrische Berechnungen für numerische Felder durchzuführen.

Parameter

- `expression`: Ein Ausdruck, der in eine Zahl aufgelöst wird.

Das Ergebnis wird im Bogenmaß angegeben. Um Grade zu erhalten, wenden Sie `$radiansToDegrees` sich an die Ausgabe.

Der Rückgabetyp ist `double` standardmäßig. Wenn die Eingabe eine 128-Bit-Dezimalzahl ist, ist die Ausgabe ebenfalls eine 128-Bit-Dezimalzahl.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$asinh` Operator verwendet wird, um den inversen hyperbolischen Sinus numerischer Werte zu berechnen.

Erstellen Sie Beispieldokumente

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 1 },
```

```
  { "_id": 3, "value": -1 }
]);
```

Beispiel für eine Abfrage

```
db.values.aggregate([
  { $project: {
    "result": { $asinh: "$value" }
  }}
]);
```

Ausgabe

```
[
  { "_id": 1, "result": 0 },
  { "_id": 2, "result": 0.881373587019543 },
  { "_id": 3, "result": -0.881373587019543 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$asinh` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('values');

  const result = await collection.aggregate([
    { $project: {
      "result": { $asinh: "$value" }
    }}
  ]).toArray();

  console.log(result);
```

```
  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['values']

    result = list(collection.aggregate([
        { '$project': {
            'result': { '$asinh': '$value' }
        }}
    ]))

    print(result)
    client.close()

if __name__ == "__main__":
    main()
```

$atan

Neu ab Version 8.0.1.

Der `$atan` Operator in Amazon DocumentDB gibt den umgekehrten Tangens (Arkustangens) eines Werts zurück.

Der Eingabeausdruck muss in einen numerischen Wert aufgelöst werden.

Das Ergebnis ist im Bogenmaß angegeben. Um Grade zu erhalten, wenden Sie `$radiansToDegrees` sich an die Ausgabe.

Der Rückgabetyp ist `double` standardmäßig. Wenn die Eingabe eine 128-Bit-Dezimalzahl ist, ist die Ausgabe ebenfalls eine 128-Bit-Dezimalzahl.

Parameter

- `expression`: Ein Ausdruck, der in eine Zahl aufgelöst wird.

Behavior

null und NaN

Beispiel	Ergebnisse
`{ $atan: NaN }`	`NaN`
`{ $atan: null }`	`null`

Wenn die Eingabe ist `null` oder das Feld, auf das verwiesen wird, fehlt, lautet das Ergebnis`null`. Eine Eingabe von `NaN` ergibt`NaN`.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$atan` Operator verwendet wird, um den Arktangens eines Werts zu berechnen.

Erstellen Sie Beispieldokumente

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 1 },
  { "_id": 3, "value": -1 }
]);
```

Beispiel für eine Abfrage

```
db.values.aggregate([
  { $project: {
    "angle": { $atan: "$value" }
  }}
]);
```

Ausgabe

```
[
  { "_id": 1, "angle": 0 },
  { "_id": 2, "angle": 0.7853981633974483 },
  { "_id": 3, "angle": -0.7853981633974483 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$atan` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('values');

  const result = await collection.aggregate([
    { $project: {
      "angle": { $atan: "$value" }
    }}
  ]).toArray();

  console.log(result);
  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
```

```
    db = client['test']
    collection = db['values']

    result = list(collection.aggregate([
        { '$project': {
            'angle': { '$atan': '$value' }
        }}
    ]))

    print(result)
    client.close()

if __name__ == "__main__":
    main()
```

$atan2

Neu ab Version 8.0.1.

Der `$atan2` Operator in Amazon DocumentDB gibt den umgekehrten Tangens (Arkustangens) von zurück`y / x`, wobei y und der erste bzw. zweite Wert `x` sind, der an den Ausdruck übergeben wird.

Beide Ausdrücke müssen in numerische Werte aufgelöst werden. Der Operator verwendet das Vorzeichen beider Argumente, um den richtigen Quadranten zu bestimmen.

Das Ergebnis wird im Bogenmaß angegeben. Um Grade zu erhalten, wenden Sie `$radiansToDegrees` sich an die Ausgabe.

Der Rückgabetyp ist `double` standardmäßig. Wenn die Eingaben 128-Bit-Dezimalzahlen sind, ist die Ausgabe ebenfalls eine 128-Bit-Dezimalzahl.

Parameter

- `expression 1`(y): Ein Ausdruck, der in eine Zahl aufgelöst wird, die die Y-Koordinate darstellt.
- `expression 2`(x): Ein Ausdruck, der in eine Zahl aufgelöst wird, die die X-Koordinate darstellt.

Behavior

null und NaN

Wenn eines der Argumente gleich ist `null` (oder auf ein fehlendes Feld verweist), ist `null` das Ergebnis Wenn eines der Argumente gleich ist`NaN`, ist das Ergebnis`NaN`. Wenn ein Argument ist `null` und das andere ist`NaN`, hat `null` Vorrang.

Beispiel	Ergebnisse
`{ $atan2: [ NaN, <value> ] }`	`NaN`
`{ $atan2: [ <value>, NaN ] }`	`NaN`
`{ $atan2: [ null, <value> ] }`	`null`
`{ $atan2: [ <value>, null ] }`	`null`
`{ $atan2: [ NaN, null ] }`	`null`
`{ $atan2: [ null, NaN ] }`	`null`

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$atan2` Operator verwendet wird, um den Arkustangens von y/x für eine Reihe von Punkten zu berechnen.

Erstellen Sie Beispieldokumente

```
db.points.insertMany([
  { "_id": 1, "y": 1, "x": 1 },
  { "_id": 2, "y": 1, "x": 0 },
  { "_id": 3, "y": -1, "x": -1 }
]);
```

Beispiel für eine Abfrage

```
db.points.aggregate([
  { $project: {
    "angle": { $atan2: ["$y", "$x"] }
  }}
]);
```

Ausgabe

```
[
  { "_id": 1, "angle": 0.7853981633974483 },
  { "_id": 2, "angle": 1.5707963267948966 },
  { "_id": 3, "angle": -2.356194490192345 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$atan2` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('points');

  const result = await collection.aggregate([
    { $project: {
      "angle": { $atan2: ["$y", "$x"] }
    }}
  ]).toArray();

  console.log(result);
  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
```

```
    db = client['test']
    collection = db['points']

    result = list(collection.aggregate([
        { '$project': {
            'angle': { '$atan2': ['$y', '$x'] }
        }}
    ]))

    print(result)
    client.close()

if __name__ == "__main__":
    main()
```

$atanh

Neu ab Version 8.0.1.

Der `$atanh` Operator in Amazon DocumentDB gibt den umgekehrten hyperbolischen Tangens (hyperbolische Bogentangente) eines Werts zurück.

Der Eingabeausdruck muss in eine Zahl im Bereich bis (einschließlich) aufgelöst werden. `-1` `1` Werte außerhalb dieses Bereichs verursachen einen Fehler.

Der Rückgabetyp ist `double` standardmäßig. Wenn die Eingabe eine 128-Bit-Dezimalzahl ist, ist die Ausgabe ebenfalls eine 128-Bit-Dezimalzahl.

Parameter

- `expression`: Ein Ausdruck, der in eine Zahl zwischen `-1` und 1 (einschließlich) aufgelöst wird.

Behavior

Null, NaN und +/- Unendlich

Beispiel	Ergebnisse
`{ $atanh: NaN }`	`NaN`
`{ $atanh: null }`	`null`

Beispiel	Ergebnisse
`{ $atanh: 1 }`	`Infinity`
`{ $atanh: -1 }`	`-Infinity`
`{ $atanh: Infinity }` oder `{ $atanh: -Infinity }`	Wirft einen Fehler aus.

Wenn die Eingabe ist `null` oder das Feld, auf das verwiesen wird, fehlt, lautet `null` das Ergebnis. Eine Eingabe von `NaN` ergibt`NaN`. Die Grenzwerte `1` `-1` `Infinity` und das `-Infinity` Produkt bzw. Jeder Wert außerhalb des `[-1, 1]` Bereichs (einschließlich`±Infinity`) verursacht einen Fehler.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$atanh` Operator verwendet wird, um den inversen hyperbolischen Tangens eines Werts zu berechnen.

Erstellen Sie Beispieldokumente

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 0.5 },
  { "_id": 3, "value": -0.5 }
]);
```

Beispiel für eine Abfrage

```
db.values.aggregate([
  { $project: {
    "result": { $atanh: "$value" }
  }}
]);
```

Ausgabe

```
[
  { "_id": 1, "result": 0 },
  { "_id": 2, "result": 0.5493061443340548 },
  { "_id": 3, "result": -0.5493061443340548 }
```

```
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$atanh` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('values');

  const result = await collection.aggregate([
    { $project: {
      "result": { $atanh: "$value" }
    }}
  ]).toArray();

  console.log(result);
  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['values']

    result = list(collection.aggregate([
        { '$project': {
```

```
            'result': { '$atanh': '$value' }
        }}
    ]))

    print(result)
    client.close()

if __name__ == "__main__":
    main()
```

$avg

Der `$avg` Aggregationsoperator in Amazon DocumentDB berechnet den Durchschnittswert des angegebenen Ausdrucks in den Dokumenten, die in die Bühne eingegeben werden. Dieser Operator ist nützlich, um den Durchschnitt eines numerischen Felds oder Ausdrucks in einer Reihe von Dokumenten zu berechnen.

Parameter

- `expression`: Der Ausdruck, der zur Berechnung des Durchschnitts verwendet werden soll. Dies kann ein Feldpfad (z. B.`"$field"`) oder ein Ausdruck (z. B.`{ $multiply: ["$field1", "$field2"] }`) sein.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$avg` Operator verwendet wird, um die durchschnittliche Punktzahl für eine Reihe von Studentendokumenten zu berechnen.

Erstellen Sie Beispieldokumente

```
db.students.insertMany([
  { name: "John", score: 85 },
  { name: "Jane", score: 92 },
  { name: "Bob", score: 78 },
  { name: "Alice", score: 90 }
]);
```

Beispiel für eine Abfrage

```
db.students.aggregate([
```

```
  { $group: {
    _id: null,
    avgScore: { $avg: "$score" }
  }}
]);
```

Ausgabe

```
[
  {
    "_id": null,
    "avgScore": 86.25
  }
]
```

Beispiel für die Verwendung von Fensteroperatoren (MongoDB Shell)

Neu ab Version 8.0.2.

Der `$avg` Operator kann auch als Fensterbediener auf der `$setWindowFields` Bühne verwendet werden. In diesem Zusammenhang gibt es den Durchschnitt des angegebenen Ausdrucks für die Dokumente in jedem Fenster zurück. Sie geben den Operator unter dem `output` Feld an und definieren optional die Fenstergrenzen mit einem `window` Dokument.

Erstellen Sie Beispieldokumente

```
db.readings.insertMany([
  { _id: 1, city: "Denver", day: 1, temp: 70 },
  { _id: 2, city: "Denver", day: 2, temp: 80 },
  { _id: 3, city: "Seattle", day: 1, temp: 60 },
  { _id: 4, city: "Seattle", day: 2, temp: 64 },
  { _id: 5, city: "Seattle", day: 3, temp: 68 }
]);
```

Beispiel für eine Abfrage

Das folgende Beispiel partitioniert die Dokumente nach`city`, sortiert jede Partition nach `day` und gibt einen laufenden Durchschnitt `temp` vom Anfang der Partition bis zum aktuellen Dokument zurück.

```
db.readings.aggregate([
  {
    $setWindowFields: {
```

```
      partitionBy: "$city",
      sortBy: { day: 1 },
      output: {
        runningAvg: {
          $avg: "$temp",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "city": "Denver", "day": 1, "temp": 70, "runningAvg": 70 },
  { "_id": 2, "city": "Denver", "day": 2, "temp": 80, "runningAvg": 75 },
  { "_id": 3, "city": "Seattle", "day": 1, "temp": 60, "runningAvg": 60 },
  { "_id": 4, "city": "Seattle", "day": 2, "temp": 64, "runningAvg": 62 },
  { "_id": 5, "city": "Seattle", "day": 3, "temp": 68, "runningAvg": 64 }
]
```

Jedes Dokument wird um den Durchschnitt `temp` innerhalb seiner Partition bis einschließlich des aktuellen Dokuments erweitert. `runningAvg`

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$avg` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten. Die folgenden Beispiele zeigen sowohl die Verwendung des Akkumulators (in`$group`) als auch die Verwendung von Fensteroperatoren (in`$setWindowFields`):

Node.js

```
const { MongoClient } = require('mongodb');

async function calculateAvgScore() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
```

```
  // Accumulator usage: average across grouped documents
  const students = db.collection('students');
  await students.insertMany([
    { name: "John", score: 85 },
    { name: "Jane", score: 92 },
    { name: "Bob", score: 78 },
    { name: "Alice", score: 90 }
  ]);
  const accumulatorResult = await students.aggregate([
    { $group: {
      _id: null,
      avgScore: { $avg: '$score' }
    }}
  ]).toArray();
  console.log('Accumulator result:', accumulatorResult);

  // Window operator usage: running average within each partition
  const readings = db.collection('readings');
  await readings.insertMany([
    { _id: 1, city: "Denver", day: 1, temp: 70 },
    { _id: 2, city: "Denver", day: 2, temp: 80 },
    { _id: 3, city: "Seattle", day: 1, temp: 60 },
    { _id: 4, city: "Seattle", day: 2, temp: 64 },
    { _id: 5, city: "Seattle", day: 3, temp: 68 }
  ]);
  const windowResult = await readings.aggregate([
    {
      $setWindowFields: {
        partitionBy: "$city",
        sortBy: { day: 1 },
        output: {
          runningAvg: {
            $avg: "$temp",
            window: { documents: ["unbounded", "current"] }
          }
        }
      }
    }
  ]).toArray();
  console.log('Window result:', windowResult);

  await client.close();
}
```

```
calculateAvgScore();
```

Python

```
from pymongo import MongoClient

def calculate_avg_score():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test

    # Accumulator usage: average across grouped documents
    students = db.students
    students.insert_many([
        { 'name': 'John', 'score': 85 },
        { 'name': 'Jane', 'score': 92 },
        { 'name': 'Bob', 'score': 78 },
        { 'name': 'Alice', 'score': 90 }
    ])
    accumulator_result = list(students.aggregate([
        { '$group': {
            '_id': None,
            'avgScore': { '$avg': '$score' }
        }}
    ]))
    print('Accumulator result:', accumulator_result)

    # Window operator usage: running average within each partition
    readings = db.readings
    readings.insert_many([
        { '_id': 1, 'city': 'Denver', 'day': 1, 'temp': 70 },
        { '_id': 2, 'city': 'Denver', 'day': 2, 'temp': 80 },
        { '_id': 3, 'city': 'Seattle', 'day': 1, 'temp': 60 },
        { '_id': 4, 'city': 'Seattle', 'day': 2, 'temp': 64 },
        { '_id': 5, 'city': 'Seattle', 'day': 3, 'temp': 68 }
    ])
    window_result = list(readings.aggregate([
        {
            '$setWindowFields': {
                'partitionBy': '$city',
                'sortBy': { 'day': 1 },
                'output': {
```

```
                    'runningAvg': {
                        '$avg': '$temp',
                        'window': { 'documents': ['unbounded', 'current'] }
                    }
                }
            }
        }
    ]))
    print('Window result:', window_result)

    client.close()

calculate_avg_score()
```

$ Binärgröße

Neu ab Version 8.0.1.

Der `$binarySize` Operator in Amazon DocumentDB gibt die Größe einer bestimmten Zeichenfolge oder eines binären Datenwerts in Byte zurück. Bei Zeichenfolgenwerten ist dies die Anzahl der UTF-8 kodierten Byte, nicht die Anzahl der Zeichen. Multi-byte UTF-8 Zeichen (wie Zeichen mit Akzenten oder CJK-Zeichen) ergeben eine Byteanzahl, die größer ist als die Anzahl der Zeichen in der Zeichenfolge.

Parameter

- `expression`: Ein Ausdruck, der in eine Zeichenfolge oder einen binären Datenwert aufgelöst wird.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$binarySize` Operator verwendet wird, um die Bytegröße von Zeichenfolgenfeldern zurückzugeben.

Erstellen Sie Beispieldokumente

```
db.docs.insertMany([
  {_id: 1, content: "hello"},
  {_id: 2, content: "Amazon DocumentDB"},
  {_id: 3, content: ""}
]);
```

Beispiel für eine Abfrage

```
db.docs.aggregate([
  { $project: { size: { $binarySize: "$content" } } }
]);
```

Ausgabe

```
[
  {_id: 1, size: 5},
  {_id: 2, size: 17},
  {_id: 3, size: 0}
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$binarySize` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  try {
    await client.connect();
    const db = client.db('test');
    const collection = db.collection('docs');
    const result = await collection.aggregate([
      { $project: { size: { $binarySize: "$content" } } }
    ]).toArray();
    console.log(result);
  } finally {
    await client.close();
  }
}
example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    try:
        db = client['test']
        collection = db['docs']
        result = list(collection.aggregate([
            {'$project': {'size': {'$binarySize': '$content'}}}
        ]))
        print(result)
    finally:
        client.close()

example()
```

$bit und

Neu ab Version 8.0.1.

Der `$bitAnd` Operator in Amazon DocumentDB führt eine bitweise UND-Operation für Integer- oder Long-Werte durch.

Parameter

- `expressions`: Ein Array mit zwei oder mehr Ausdrücken, das in Ganzzahlen oder Long-Werte aufgelöst werden kann.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$bitAnd` Operator verwendet wird, um bitweise UND für zwei Felder auszuführen.

Erstellen Sie Beispieldokumente

```
db.flags.insertMany([
```

```
  {_id: 1, a: 13, b: 10},
  {_id: 2, a: 7, b: 5},
  {_id: 3, a: 15, b: 9}
]);
```

Beispiel für eine Abfrage

```
db.flags.aggregate([
  { $project: { result: { $bitAnd: ["$a", "$b"] } } }
]);
```

Ausgabe

```
[
  {_id: 1, result: 8},
  {_id: 2, result: 5},
  {_id: 3, result: 9}
]
```

Binär: 13 (1101) UND 10 (1010) = 8 (1000); 7 (0111) UND 5 (0101) = 5 (0101); 15 (1111) UND 9 (1001) = 9 (1001).

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$bitAnd` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  try {
    await client.connect();
    const db = client.db('test');
    const collection = db.collection('flags');
    const result = await collection.aggregate([
      { $project: { result: { $bitAnd: ["$a", "$b"] } } }
```

```
    ]).toArray();
    console.log(result);
  } finally {
    await client.close();
  }
}
example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    try:
        db = client['test']
        collection = db['flags']
        result = list(collection.aggregate([
            {'$project': {'result': {'$bitAnd': ['$a', '$b']}}}
        ]))
        print(result)
    finally:
        client.close()

example()
```

$ Bit nicht

Neu ab Version 8.0.1.

Der `$bitNot` Operator in Amazon DocumentDB führt eine bitweise NOT-Operation für eine Ganzzahl oder einen langen Wert durch und gibt das bitweise Komplement zurück. Für eine bestimmte Ganzzahl oder einen langen Wert von n ist das Ergebnis - (n+1).

Parameter

- `expression`: Ein Ausdruck, der in eine Ganzzahl oder einen Wert vom Typ Long aufgelöst wird. Gibt das bitweise Komplement zurück, das - (n+1) für eine bestimmte Ganzzahl oder einen langen n ist.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$bitNot` Operator verwendet wird, um das bitweise Komplement von Ganzzahlwerten zu berechnen.

Erstellen Sie Beispieldokumente

```
db.numbers.insertMany([
  {_id: 1, value: 0},
  {_id: 2, value: 5},
  {_id: 3, value: -3}
]);
```

Beispiel für eine Abfrage

```
db.numbers.aggregate([
  { $project: { result: { $bitNot: "$value" } } }
]);
```

Ausgabe

```
[
  {_id: 1, result: -1},
  {_id: 2, result: -6},
  {_id: 3, result: 2}
]
```

Im Binärformat: NICHT 0 = -1; NICHT 5 = -6; NICHT -3 = 2.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$bitNot` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
```

```
  try {
    await client.connect();
    const db = client.db('test');
    const collection = db.collection('numbers');
    const result = await collection.aggregate([
      { $project: { result: { $bitNot: "$value" } } }
    ]).toArray();
    console.log(result);
  } finally {
    await client.close();
  }
}
example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    try:
        db = client['test']
        collection = db['numbers']
        result = list(collection.aggregate([
            {'$project': {'result': {'$bitNot': '$value'}}}
        ]))
        print(result)
    finally:
        client.close()

example()
```

$ Bit oder

Neu ab Version 8.0.1.

Der `$bitOr` Operator in Amazon DocumentDB führt eine bitweise ODER-Operation für Integer- oder Long-Werte durch.

Parameter

- `expressions`: Ein Array mit zwei oder mehr Ausdrücken, das in Ganzzahlen oder Long-Werte aufgelöst werden kann.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$bitOr` Operator verwendet wird, um bitweises ODER für zwei Felder auszuführen.

Erstellen Sie Beispieldokumente

```
db.flags.insertMany([
  {_id: 1, a: 13, b: 10},
  {_id: 2, a: 7, b: 5},
  {_id: 3, a: 15, b: 9}
]);
```

Beispiel für eine Abfrage

```
db.flags.aggregate([
  { $project: { result: { $bitOr: ["$a", "$b"] } } }
]);
```

Ausgabe

```
[
  {_id: 1, result: 15},
  {_id: 2, result: 7},
  {_id: 3, result: 15}
]
```

Binär: 13 (1101) ODER 10 (1010) = 15 (1111); 7 (0111) ODER 5 (0101) = 7 (0111); 15 (1111) ODER 9 (1001) = 15 (1111).

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$bitOr` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');
```

```
async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  try {
    await client.connect();
    const db = client.db('test');
    const collection = db.collection('flags');
    const result = await collection.aggregate([
      { $project: { result: { $bitOr: ["$a", "$b"] } } }
    ]).toArray();
    console.log(result);
  } finally {
    await client.close();
  }
}
example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    try:
        db = client['test']
        collection = db['flags']
        result = list(collection.aggregate([
            {'$project': {'result': {'$bitOr': ['$a', '$b']}}}
        ]))
        print(result)
    finally:
        client.close()

example()
```

$ bitX oder

Neu ab Version 8.0.1.

Der `$bitXor` Operator in Amazon DocumentDB führt eine bitweise XOR-Operation für Integer- oder Long-Werte durch.

Parameter

- `expressions`: Ein Array aus zwei oder mehr Ausdrücken, das in Ganzzahlen oder Long-Werte aufgelöst werden kann.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$bitXor` Operator verwendet wird, um bitweises XOR für zwei Felder auszuführen.

Erstellen Sie Beispieldokumente

```
db.flags.insertMany([
  {_id: 1, a: 13, b: 10},
  {_id: 2, a: 7, b: 5},
  {_id: 3, a: 15, b: 9}
]);
```

Beispiel für eine Abfrage

```
db.flags.aggregate([
  { $project: { result: { $bitXor: ["$a", "$b"] } } }
]);
```

Ausgabe

```
[
  {_id: 1, result: 7},
  {_id: 2, result: 2},
  {_id: 3, result: 6}
]
```

Binär: 13 (1101) XOR 10 (1010) = 7 (0111); 7 (0111) XOR 5 (0101) = 2 (0010); 15 (1111) XOR 9 (1001) = 6 (0110).

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$bitXor` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  try {
    await client.connect();
    const db = client.db('test');
    const collection = db.collection('flags');
    const result = await collection.aggregate([
      { $project: { result: { $bitXor: ["$a", "$b"] } } }
    ]).toArray();
    console.log(result);
  } finally {
    await client.close();
  }
}
example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    try:
        db = client['test']
        collection = db['flags']
        result = list(collection.aggregate([
            {'$project': {'result': {'$bitXor': ['$a', '$b']}}}
        ]))
        print(result)
    finally:
```

```
        client.close()

example()
```

$ unten

Neu ab Version 8.0.1.

Verwenden Sie den `$bottom` Akkumulator in der `$group` Phase, um das Dokument mit dem niedrigsten Rang pro Gruppe gemäß einer bestimmten Sortierreihenfolge zurückzugeben.

Parameter

- `sortBy`: Ein Dokument, das die Sortierreihenfolge angibt. Wird 1 für aufsteigend oder `-1` für absteigend verwendet.
- `output`: Ein Ausdruck, der die Felder angibt, die aus dem untersten Dokument zurückgegeben werden sollen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$bottom` Akkumulator verwendet wird, um den niedrigsten Umsatz (niedrigste Menge) pro Artikel in einer Verkaufskollektion zu ermitteln.

Erstellen Sie Beispieldokumente

```
db.sales.insertMany([
  { item: "abc", quantity: 10, price: 5 },
  { item: "abc", quantity: 5, price: 8 },
  { item: "xyz", quantity: 15, price: 3 },
  { item: "xyz", quantity: 7, price: 6 }
])
```

Beispiel für eine Abfrage

```
db.sales.aggregate([
  { $group: { _id: "$item", bottomSale: { $bottom: { sortBy: { quantity: -1 }, output:
 { quantity: "$quantity", price: "$price" } } } } }
])
```

Ausgabe

```
[
  { "_id": "xyz", "bottomSale": { "quantity": 7, "price": 6 } },
  { "_id": "abc", "bottomSale": { "quantity": 5, "price": 8 } }
]
```

Beispiel für die Verwendung von Fensteroperatoren (MongoDB Shell)

Neu ab Version 8.0.2.

Der `$bottom` Operator kann auch als Fensterbediener auf der `$setWindowFields` Bühne verwendet werden. In diesem Zusammenhang gibt es das Dokument `output` aus dem Dokument zurück, das unter den Dokumenten in jedem Fenster an unterster Stelle steht (entsprechend den Angaben des Operators`sortBy`). Sie geben den Operator unter dem `output` Feld an und definieren optional die Fenstergrenzen mit einem `window` Dokument.

> **Note**
>
> Bei Verwendung als Fensteroperator in `$setWindowFields` `$bottom` ist er auf 100 MB an Zwischendaten begrenzt. Eine Operation, die dieses Limit überschreitet, gibt einen Fehler zurück.

Erstellen Sie Beispieldokumente

```
db.matchScores.insertMany([
  { _id: 1, player: "Alice", round: 1, score: 20 },
  { _id: 2, player: "Alice", round: 2, score: 35 },
  { _id: 3, player: "Alice", round: 3, score: 28 },
  { _id: 4, player: "Bob", round: 1, score: 15 },
  { _id: 5, player: "Bob", round: 2, score: 40 }
]);
```

Beispiel für eine Abfrage

Das folgende Beispiel partitioniert die Dokumente nach`player`, sortiert jede Partition nach und gibt die Runde mit der niedrigsten Punktzahl zurück`round`, die vom Anfang der Partition bis zum aktuellen Dokument gesehen wurde.

```
db.matchScores.aggregate([
  {
```

```
    $setWindowFields: {
      partitionBy: "$player",
      sortBy: { round: 1 },
      output: {
        worstRoundSoFar: {
          $bottom: { sortBy: { score: -1 }, output: { round: "$round", score:
 "$score" } },
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "player": "Alice", "round": 1, "score": 20, "worstRoundSoFar": { "round":
 1, "score": 20 } },
  { "_id": 2, "player": "Alice", "round": 2, "score": 35, "worstRoundSoFar": { "round":
 1, "score": 20 } },
  { "_id": 3, "player": "Alice", "round": 3, "score": 28, "worstRoundSoFar": { "round":
 1, "score": 20 } },
  { "_id": 4, "player": "Bob", "round": 1, "score": 15, "worstRoundSoFar": { "round":
 1, "score": 15 } },
  { "_id": 5, "player": "Bob", "round": 2, "score": 40, "worstRoundSoFar": { "round":
 1, "score": 15 } }
]
```

Jedes Dokument wird um die Runde mit `worstRoundSoFar` der niedrigsten Punktzahl innerhalb der Partition bis einschließlich des aktuellen Dokuments erweitert.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$bottom` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten. Die folgenden Beispiele zeigen sowohl die Verwendung des Akkumulators (in`$group`) als auch die Verwendung von Fensteroperatoren (in`$setWindowFields`):

Node.js

```
const { MongoClient } = require('mongodb');
```

```
async function example() {
  const uri = 'mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false';
  const client = new MongoClient(uri);

  try {
    await client.connect();

    const db = client.db('test');

    // Accumulator usage: bottom document per group
    const sales = db.collection('sales');
    await sales.insertMany([
      { item: "abc", quantity: 10, price: 5 },
      { item: "abc", quantity: 5, price: 8 },
      { item: "xyz", quantity: 15, price: 3 },
      { item: "xyz", quantity: 7, price: 6 }
    ]);
    const accumulatorResult = await sales.aggregate([
      { $group: { _id: "$item", bottomSale: { $bottom: { sortBy: { quantity: -1 },
 output: { quantity: "$quantity", price: "$price" } } } } }
    ]).toArray();
    console.log('Accumulator result:', accumulatorResult);

    // Window operator usage: worst round so far within each partition
    const matchScores = db.collection('matchScores');
    await matchScores.insertMany([
      { _id: 1, player: "Alice", round: 1, score: 20 },
      { _id: 2, player: "Alice", round: 2, score: 35 },
      { _id: 3, player: "Alice", round: 3, score: 28 },
      { _id: 4, player: "Bob", round: 1, score: 15 },
      { _id: 5, player: "Bob", round: 2, score: 40 }
    ]);
    const windowResult = await matchScores.aggregate([
      {
        $setWindowFields: {
          partitionBy: "$player",
          sortBy: { round: 1 },
          output: {
            worstRoundSoFar: {
              $bottom: { sortBy: { score: -1 }, output: { round: "$round", score:
 "$score" } },
```

```
              window: { documents: ["unbounded", "current"] }
            }
          }
        }
      }
    ]).toArray();
    console.log('Window result:', windowResult);

  } catch (error) {
    console.error('Error:', error);
  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient
from pprint import pprint

def example():
    client = None
    try:
        client = MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

        db = client['test']

        # Accumulator usage: bottom document per group
        sales = db['sales']
        sales.insert_many([
            { 'item': 'abc', 'quantity': 10, 'price': 5 },
            { 'item': 'abc', 'quantity': 5, 'price': 8 },
            { 'item': 'xyz', 'quantity': 15, 'price': 3 },
            { 'item': 'xyz', 'quantity': 7, 'price': 6 }
        ])
        accumulator_result = list(sales.aggregate([
            { '$group': { '_id': '$item', 'bottomSale': { '$bottom':
 { 'sortBy': { 'quantity': -1 }, 'output': { 'quantity': '$quantity', 'price':
 '$price' } } } } }
```

```
        ]))
        pprint(accumulator_result)

        # Window operator usage: worst round so far within each partition
        match_scores = db['matchScores']
        match_scores.insert_many([
            { '_id': 1, 'player': 'Alice', 'round': 1, 'score': 20 },
            { '_id': 2, 'player': 'Alice', 'round': 2, 'score': 35 },
            { '_id': 3, 'player': 'Alice', 'round': 3, 'score': 28 },
            { '_id': 4, 'player': 'Bob', 'round': 1, 'score': 15 },
            { '_id': 5, 'player': 'Bob', 'round': 2, 'score': 40 }
        ])
        window_result = list(match_scores.aggregate([
            {
                '$setWindowFields': {
                    'partitionBy': '$player',
                    'sortBy': { 'round': 1 },
                    'output': {
                        'worstRoundSoFar': {
                            '$bottom': { 'sortBy': { 'score': -1 }, 'output':
 { 'round': '$round', 'score': '$score' } },
                            'window': { 'documents': ['unbounded', 'current'] }
                        }
                    }
                }
            }
        ]))
        pprint(window_result)

    except Exception as e:
        print(f"An error occurred: {e}")

    finally:
        if client:
            client.close()

example()
```

$ unten N

Neu ab Version 8.0.1.

Verwenden Sie den `$bottomN` Akkumulator in der `$group` Phase, um die untersten N Elemente innerhalb einer Gruppe gemäß einer bestimmten Sortierreihenfolge zurückzugeben. Wenn eine Gruppe weniger als N Elemente enthält, werden alle Elemente in der Gruppe `$bottomN` zurückgegeben.

Parameter

- n: Eine positive Ganzzahl oder ein Ausdruck, der in eins aufgelöst wird und angibt, wie viele unterste Ergebnisse pro Gruppe zurückgegeben werden sollen.
- `sortBy`: Ein Dokument, das die Sortierreihenfolge angibt. Wird 1 für aufsteigend oder `-1` für absteigend verwendet.
- `output`: Ein Ausdruck, der die Felder angibt, die von jedem der unteren N Dokumente zurückgegeben werden sollen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$bottomN` Akkumulator verwendet wird, um die beiden niedrigsten Verkäufe (niedrigste Menge) pro Artikel in einer Verkaufskollektion zu ermitteln.

Erstellen Sie Beispieldokumente

```
db.sales.insertMany([
  { item: "abc", quantity: 10, price: 5 },
  { item: "abc", quantity: 7, price: 8 },
  { item: "abc", quantity: 5, price: 10 },
  { item: "xyz", quantity: 15, price: 3 },
  { item: "xyz", quantity: 9, price: 6 },
  { item: "xyz", quantity: 3, price: 12 }
])
```

Beispiel für eine Abfrage

```
db.sales.aggregate([
  { $group: { _id: "$item", bottomTwoSales: { $bottomN: { n: 2, sortBy: { quantity:
 -1 }, output: { quantity: "$quantity", price: "$price" } } } } }
])
```

Ausgabe

```
[
```

```
  { "_id": "xyz", "bottomTwoSales": [{ "quantity": 9, "price": 6 }, { "quantity": 3,
 "price": 12 }] },
  { "_id": "abc", "bottomTwoSales": [{ "quantity": 7, "price": 8 }, { "quantity": 5,
 "price": 10 }] }
]
```

Beispiel für die Verwendung von Fensteroperatoren (MongoDB Shell)

Neu ab Version 8.0.2.

Der `$bottomN` Operator kann auch als Fensterbediener auf der `$setWindowFields` Bühne verwendet werden. In diesem Zusammenhang gibt er unter den Dokumenten in jedem Fenster eine Reihe `output` von N Dokumenten (entsprechend den Angaben des Operators`sortBy`) von unten zurück. Sie geben den Operator unter dem `output` Feld an und definieren optional die Fenstergrenzen mit einem `window` Dokument.

> **Note**
>
> Bei Verwendung als Fensteroperator in `$setWindowFields` `$bottomN` ist er auf 100 MB an Zwischendaten begrenzt. Eine Operation, die dieses Limit überschreitet, gibt einen Fehler zurück.

Erstellen Sie Beispieldokumente

```
db.matchScores.insertMany([
  { _id: 1, player: "Alice", round: 1, score: 20 },
  { _id: 2, player: "Alice", round: 2, score: 35 },
  { _id: 3, player: "Alice", round: 3, score: 28 },
  { _id: 4, player: "Bob", round: 1, score: 15 },
  { _id: 5, player: "Bob", round: 2, score: 40 }
]);
```

Beispiel für eine Abfrage

Das folgende Beispiel partitioniert die Dokumente nach`player`, sortiert jede Partition nach `round` und gibt die beiden Runden mit der niedrigsten Punktzahl vom Anfang der Partition bis zum aktuellen Dokument zurück.

```
db.matchScores.aggregate([
```

```
  {
    $setWindowFields: {
      partitionBy: "$player",
      sortBy: { round: 1 },
      output: {
        bottomTwoRoundsSoFar: {
          $bottomN: { n: 2, sortBy: { score: -1 }, output: { round: "$round", score:
 "$score" } },
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "player": "Alice", "round": 1, "score": 20, "bottomTwoRoundsSoFar":
 [{ "round": 1, "score": 20 }] },
  { "_id": 2, "player": "Alice", "round": 2, "score": 35, "bottomTwoRoundsSoFar":
 [{ "round": 2, "score": 35 }, { "round": 1, "score": 20 }] },
  { "_id": 3, "player": "Alice", "round": 3, "score": 28, "bottomTwoRoundsSoFar":
 [{ "round": 3, "score": 28 }, { "round": 1, "score": 20 }] },
  { "_id": 4, "player": "Bob", "round": 1, "score": 15, "bottomTwoRoundsSoFar":
 [{ "round": 1, "score": 15 }] },
  { "_id": 5, "player": "Bob", "round": 2, "score": 40, "bottomTwoRoundsSoFar":
 [{ "round": 2, "score": 40 }, { "round": 1, "score": 15 }] }
]
```

Jedes Dokument wird um die beiden Runden mit `bottomTwoRoundsSoFar` der niedrigsten Punktzahl innerhalb der Partition bis einschließlich des aktuellen Dokuments erweitert. Wenn das Fenster weniger als zwei Dokumente enthält, werden alle zurückgegeben.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$bottomN` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten. Die folgenden Beispiele zeigen sowohl die Verwendung des Akkumulators (in`$group`) als auch die Verwendung von Fensteroperatoren (in`$setWindowFields`):

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const uri = 'mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false';
  const client = new MongoClient(uri);

  try {
    await client.connect();

    const db = client.db('test');

    // Accumulator usage: bottom N documents per group
    const sales = db.collection('sales');
    await sales.insertMany([
      { item: "abc", quantity: 10, price: 5 },
      { item: "abc", quantity: 7, price: 8 },
      { item: "abc", quantity: 5, price: 10 },
      { item: "xyz", quantity: 15, price: 3 },
      { item: "xyz", quantity: 9, price: 6 },
      { item: "xyz", quantity: 3, price: 12 }
    ]);
    const accumulatorResult = await sales.aggregate([
      { $group: { _id: "$item", bottomTwoSales: { $bottomN: { n: 2, sortBy:
 { quantity: -1 }, output: { quantity: "$quantity", price: "$price" } } } } }
    ]).toArray();
    console.log('Accumulator result:', accumulatorResult);

    // Window operator usage: bottom two rounds so far within each partition
    const matchScores = db.collection('matchScores');
    await matchScores.insertMany([
      { _id: 1, player: "Alice", round: 1, score: 20 },
      { _id: 2, player: "Alice", round: 2, score: 35 },
      { _id: 3, player: "Alice", round: 3, score: 28 },
      { _id: 4, player: "Bob", round: 1, score: 15 },
      { _id: 5, player: "Bob", round: 2, score: 40 }
    ]);
    const windowResult = await matchScores.aggregate([
      {
        $setWindowFields: {
          partitionBy: "$player",
```

```
          sortBy: { round: 1 },
          output: {
            bottomTwoRoundsSoFar: {
              $bottomN: { n: 2, sortBy: { score: -1 }, output: { round: "$round",
 score: "$score" } },
              window: { documents: ["unbounded", "current"] }
            }
          }
        }
      }
    ]).toArray();
    console.log('Window result:', windowResult);

  } catch (error) {
    console.error('Error:', error);
  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient
from pprint import pprint

def example():
    client = None
    try:
        client = MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

        db = client['test']

        # Accumulator usage: bottom N documents per group
        sales = db['sales']
        sales.insert_many([
            { 'item': 'abc', 'quantity': 10, 'price': 5 },
            { 'item': 'abc', 'quantity': 7, 'price': 8 },
            { 'item': 'abc', 'quantity': 5, 'price': 10 },
            { 'item': 'xyz', 'quantity': 15, 'price': 3 },
```

```
            { 'item': 'xyz', 'quantity': 9, 'price': 6 },
            { 'item': 'xyz', 'quantity': 3, 'price': 12 }
        ])
        accumulator_result = list(sales.aggregate([
            { '$group': { '_id': '$item', 'bottomTwoSales': { '$bottomN': { 'n':
2, 'sortBy': { 'quantity': -1 }, 'output': { 'quantity': '$quantity', 'price':
'$price' } } } } }
        ]))
        pprint(accumulator_result)

        # Window operator usage: bottom two rounds so far within each partition
        match_scores = db['matchScores']
        match_scores.insert_many([
            { '_id': 1, 'player': 'Alice', 'round': 1, 'score': 20 },
            { '_id': 2, 'player': 'Alice', 'round': 2, 'score': 35 },
            { '_id': 3, 'player': 'Alice', 'round': 3, 'score': 28 },
            { '_id': 4, 'player': 'Bob', 'round': 1, 'score': 15 },
            { '_id': 5, 'player': 'Bob', 'round': 2, 'score': 40 }
        ])
        window_result = list(match_scores.aggregate([
            {
                '$setWindowFields': {
                    'partitionBy': '$player',
                    'sortBy': { 'round': 1 },
                    'output': {
                        'bottomTwoRoundsSoFar': {
                            '$bottomN': { 'n': 2, 'sortBy': { 'score': -1 },
'output': { 'round': '$round', 'score': '$score' } },
                            'window': { 'documents': ['unbounded', 'current'] }
                        }
                    }
                }
            }
        ]))
        pprint(window_result)

    except Exception as e:
        print(f"An error occurred: {e}")

    finally:
        if client:
            client.close()
```

```
example()
```

$ bsonGröße

Neu ab Version 8.0.1.

Der `$bsonSize` Operator in Amazon DocumentDB gibt die Größe eines Dokuments in Byte zurück, wenn es als BSON codiert ist.

Parameter

- `expression`: Ein Ausdruck, der in ein Dokument oder Objekt aufgelöst wird.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$bsonSize` Operator verwendet wird, um die BSON-Größe jedes Dokuments zurückzugeben.

Erstellen Sie Beispieldokumente

```
db.items.insertMany([
  {_id: 1, name: "a", tags: ["x"]},
  {_id: 2, name: "ab", tags: ["x", "y", "z"]},
  {_id: 3, name: "abc", tags: []}
]);
```

Beispiel für eine Abfrage

```
db.items.aggregate([
  { $project: { docSize: { $bsonSize: "$$ROOT" } } }
]);
```

Ausgabe

```
[
  {_id: 1, docSize: 46},
  {_id: 2, docSize: 65},
  {_id: 3, docSize: 39}
]
```

> **Note**
>
> Die genauen Größen hängen von der BSON-Codierung der Dokumentfelder ab.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$bsonSize` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  try {
    await client.connect();
    const db = client.db('test');
    const collection = db.collection('items');
    const result = await collection.aggregate([
      { $project: { docSize: { $bsonSize: "$$ROOT" } } }
    ]).toArray();
    console.log(result);
  } finally {
    await client.close();
  }
}
example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    try:
        db = client['test']
```

```
        collection = db['items']
        result = list(collection.aggregate([
            {'$project': {'docSize': {'$bsonSize': '$$ROOT'}}}
        ]))
        print(result)
    finally:
        client.close()

example()
```

$bucket

Neu ab Version 8.0

Wird von Elastic Cluster nicht unterstützt.

Die `$bucket` Aggregationsphase in Amazon DocumentDB ermöglicht es Ihnen, Eingabedokumente auf der Grundlage eines bestimmten Ausdrucks und der Bucket-Grenzen in Buckets zu gruppieren. Dies kann nützlich sein, um Daten zu analysieren, die in bestimmte Wertebereiche oder Kategorien fallen.

Parameter

- `groupBy`(erforderlich): Der Ausdruck, der den Wert angibt, nach dem gruppiert werden soll.
- `boundaries`(erforderlich): Ein Array mit doppelten Werten, die die Bucket-Grenzen definieren. Dokumente werden Buckets auf der Grundlage des `groupBy` Ausdruckswerts zugewiesen, der innerhalb der angegebenen Grenzen liegt.
- `default`(optional): Ein Literalwert, der für Dokumente ausgegeben wird, deren `groupBy` Ausdruckswert unter keine der angegebenen Grenzen fällt.
- `output`(optional): Ein Objekt, das die Informationen angibt, die für jeden Bucket ausgegeben werden sollen. Sie können Akkumulatoroperatoren wie`$sum`,, und verwenden `$avg``$min`, `$max` um Aggregationen für jeden Bucket zu berechnen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie die `$bucket` Phase verwendet wird, um Verkaufsdaten nach Preisklassen zu gruppieren.

Erstellen Sie Beispieldokumente

```
db.sales.insertMany([
  { item: "abc", price: 10, quantity: 2, date: new Date("2020-09-01") },
  { item: "def", price: 20, quantity: 1, date: new Date("2020-10-01") },
  { item: "ghi", price: 5, quantity: 3, date: new Date("2020-11-01") },
  { item: "jkl", price: 15, quantity: 2, date: new Date("2020-12-01") },
  { item: "mno", price: 25, quantity: 1, date: new Date("2021-01-01") }
]);
```

Beispiel für eine Abfrage

```
db.sales.aggregate([
  {
    $bucket: {
      groupBy: "$price",
      boundaries: [0, 10, 20, 30],
      default: "Other",
      output: {
        "count": { $sum: 1 },
        "totalQuantity": { $sum: "$quantity" }
      }
    }
  },
  {
    $sort: { _id: 1 }
  }
])
```

Ausgabe

```
[
  { _id: 0, count: 1, totalQuantity: 3 },
  { _id: 10, count: 2, totalQuantity: 4 },
  { _id: 20, count: 2, totalQuantity: 2 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$bucket` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const sales = db.collection('sales');

  const result = await sales.aggregate([
    {
      $bucket: {
        groupBy: "$price",
        boundaries: [0, 10, 20, 30],
        default: "Other",
        output: {
          "count": { $sum: 1 },
          "totalQuantity": { $sum: "$quantity" }
        }
      }
    },
    {
      $sort: { _id: 1 }
    }
  ]).toArray();

  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&lsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
```

```
    sales = db['sales']

    result = list(sales.aggregate([
        {
            '$bucket': {
                'groupBy': '$price',
                'boundaries': [0, 10, 20, 30],
                'default': 'Other',
                'output': {
                    'count': {'$sum': 1},
                    'totalQuantity': {'$sum': '$quantity'}
                }
            }
        },
        {
            "$sort": { "_id": 1 }
        }
    ]))

    print(result)
    client.close()

example()
```

$bucketAuto

Neu ab Version 8.0.2.

Die `$bucketAuto` Aggregationsphase in Amazon DocumentDB gruppiert Eingabedokumente automatisch auf der Grundlage eines angegebenen Ausdrucks in eine bestimmte Anzahl von Buckets. Im Gegensatz `$bucket` dazu, wo Sie die Bucket-Grenzen explizit definieren, `$bucketAuto` werden die Grenzen automatisch bestimmt, sodass die Dokumente so gleichmäßig wie möglich auf die angeforderte Anzahl von Buckets verteilt werden. Dies kann nützlich sein, wenn Sie Daten in eine feste Anzahl von Gruppen einteilen möchten, ohne die Verteilung der Werte im Voraus zu kennen.

Parameter

- `groupBy`(erforderlich): Der Ausdruck, der den Wert angibt, nach dem gruppiert werden soll.

- `buckets`(erforderlich): Eine positive Ganzzahl, die die Anzahl der Buckets angibt, in die die Dokumente gruppiert sind. Abhängig von den Eingabedokumenten und dem `groupBy` Ausdruck kann die tatsächliche Anzahl der Buckets unter der angegebenen Anzahl liegen.
- `output`(optional): Ein Objekt, das die Informationen angibt, die für jeden Bucket ausgegeben werden sollen. Sie können Akkumulatoroperatoren wie`$sum`,, und verwenden `$avg``$min`, `$max` um Aggregationen für jeden Bucket zu berechnen. Wenn dieser Parameter weggelassen wird, gibt jeder Bucket standardmäßig ein `count` Feld aus.
- `granularity`(optional): Eine Zeichenfolge, die die bevorzugte Zahlenreihe angibt, die für die Bucket-Grenzen verwendet werden soll. Dadurch werden die berechneten Grenzen einer numerischen Standardfolge zugeordnet. Dieser Parameter ist nur verfügbar, wenn alle `groupBy` Werte numerisch sind und keiner davon. `NaN` Gültige Werte sind: `R5``R10`,`R20`, `R40``R80`,`1-2-5`,`E6`,`E12`,`E24`,`E48`,`E96`,`E192`, und`POWERSOF2`.

Ausgabe

Jeder Bucket wird als separates Dokument zurückgegeben, das ein `_id` Objekt enthält, das die Grenzen des Buckets beschreibt:

- `_id.min`: Die inklusive Untergrenze des Buckets.
- `_id.max`: Die obere Grenze des Eimers. Diese Grenze gilt exklusiv für jeden Bereich, mit Ausnahme des letzten Buckets in der Reihe, für den sie inklusiv ist.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie die `$bucketAuto` Phase verwendet wird, um Verkaufsdaten nach Preis in Gruppen zu gruppieren.

Erstellen Sie Beispieldokumente

```
db.sales.insertMany([
  { item: "abc", price: 10, quantity: 2, date: new Date("2020-09-01") },
  { item: "def", price: 20, quantity: 1, date: new Date("2020-10-01") },
  { item: "ghi", price: 5, quantity: 3, date: new Date("2020-11-01") },
  { item: "jkl", price: 15, quantity: 2, date: new Date("2020-12-01") },
  { item: "mno", price: 25, quantity: 1, date: new Date("2021-01-01") }
]);
```

Beispiel für eine Abfrage

Mit dem folgenden Befehl werden die Beispieldokumente nach Preis in drei Gruppen gruppiert:

```
db.sales.aggregate([
  {
    $bucketAuto: {
      groupBy: "$price",
      buckets: 3,
      output: {
        "count": { $sum: 1 },
        "totalQuantity": { $sum: "$quantity" }
      }
    }
  }
])
```

Ausgabe

Der Befehl gibt die folgende Ausgabe zurück:

```
[
  { _id: { min: 5, max: 15 }, count: 2, totalQuantity: 5 },
  { _id: { min: 15, max: 25 }, count: 2, totalQuantity: 3 },
  { _id: { min: 25, max: 25 }, count: 1, totalQuantity: 1 }
]
```

Abfragebeispiel mit Granularität

Das folgende Beispiel fügt die `granularity` Option hinzu, die berechneten Bucket-Grenzen mit Potenzen von 2 zu verknüpfen.

```
db.sales.aggregate([
  {
    $bucketAuto: {
      groupBy: "$price",
      buckets: 2,
      granularity: "POWERSOF2",
      output: {
        "count": { $sum: 1 },
        "totalQuantity": { $sum: "$quantity" }
      }
    }
  }
```

```
])
```

Ausgabe

Der Befehl gibt die folgende Ausgabe zurück:

```
[
  { _id: { min: 4, max: 16 }, count: 3, totalQuantity: 7 },
  { _id: { min: 16, max: 32 }, count: 2, totalQuantity: 2 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$bucketAuto` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const sales = db.collection('sales');

  const result = await sales.aggregate([
    {
      $bucketAuto: {
        groupBy: "$price",
        buckets: 3,
        output: {
          "count": { $sum: 1 },
          "totalQuantity": { $sum: "$quantity" }
        }
      }
    }
  ]).toArray();

  console.log(result);
  client.close();
```

```
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    sales = db['sales']

    result = list(sales.aggregate([
        {
            '$bucketAuto': {
                'groupBy': '$price',
                'buckets': 3,
                'output': {
                    'count': {'$sum': 1},
                    'totalQuantity': {'$sum': '$quantity'}
                }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$ceil

Neu ab Version 4.0

Der `$ceil` Operator in Amazon DocumentDB rundet wie in MongoDB eine Zahl auf die nächste Ganzzahl auf. Dies ist nützlich, wenn Sie mathematische Operationen an numerischen Feldern ausführen und sicherstellen müssen, dass das Ergebnis eine ganze Zahl ist.

Parameter

- `expression`: Der numerische Ausdruck, der aufgerundet werden soll.

Beispiel (MongoDB Shell)

Dieses Beispiel zeigt, wie der `$ceil` Operator verwendet wird, um ein numerisches Feld aufzurunden.

Erstellen Sie Beispieldokumente

```
db.numbers.insertMany([
  { "_id": 1, "value": 3.14 },
  { "_id": 2, "value": -2.7 },
  { "_id": 3, "value": 0 }
])
```

Beispiel für eine Abfrage

```
db.numbers.aggregate([
  { $project: {
    "roundedUp": { $ceil: "$value" }
  }}
])
```

Ausgabe

```
{ "_id": 1, "roundedUp": 4 }
{ "_id": 2, "roundedUp": -2 }
{ "_id": 3, "roundedUp": 0 }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$ceil` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
```

```
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('numbers');

  const result = await collection.aggregate([
    { $project: {
      "roundedUp": { $ceil: "$value" }
    }}
  ]).toArray();

  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.numbers

    result = list(collection.aggregate([
        { '$project': {
            "roundedUp": { '$ceil': "$value" }
        }}
    ]))

    print(result)
    client.close()

example()
```

$ Stream ändern

Wird von Elastic Cluster nicht unterstützt.

In der `$changeStream` Aggregationsphase wird ein Change-Stream-Cursor geöffnet, um Änderungen an einer Sammlung in Echtzeit zu überwachen. Sie gibt Änderungsereignisdokumente zurück, wenn Einfüge-, Aktualisierungs-, Ersetzungs- oder Löschvorgänge ausgeführt werden.

Parameter

- `fullDocument`: Gibt an, ob bei Aktualisierungsvorgängen das gesamte Dokument zurückgegeben werden soll. Die Optionen lauten `default` oder `updateLookup`.
- `resumeAfter`: Optional. Setzen Sie das Token fort, um von einem bestimmten Punkt im Change-Stream aus fortzufahren.
- `startAtOperationTime`: Optional. Zeitstempel, ab dem der Change-Stream gestartet werden soll.
- `allChangesForCluster`: Optional. Boolescher Wert: Wann`true`, überwacht alle Änderungen im Cluster (für die Admin-Datenbank). Wenn `false` (Standardeinstellung), überwacht nur die angegebene Sammlung.
- `showExpandedEvents`: Optional. Boolescher Wert: Wenn`true`, schließt DDL-Ereignisbenachrichtigungen (Expanded Data Definition Language) wie `createCollection` und `createIndexes` in den Change-Stream ein. Wenn `false` (Standardeinstellung), werden diese Ereignisse nicht zurückgegeben. Verfügbar in Amazon DocumentDB 8.0 ab Engine-Version 8.0.2.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie die `$changeStream` Bühne verwendet wird, um Änderungen an einer Sammlung zu überwachen.

Beispiel für eine Abfrage

```
// Open change stream first
const changeStream = db.inventory.aggregate([
  { $changeStream: { fullDocument: "updateLookup" } }
]);

// In another session, insert a document
db.inventory.insertOne({ _id: 1, item: "Widget", qty: 10 });
```

```
// Back in the first session, read the change event
if (changeStream.hasNext()) {
  print(tojson(changeStream.next()));
}
```

Ausgabe

```
{
  _id: { _data: '...' },
  operationType: 'insert',
  clusterTime: Timestamp(1, 1234567890),
  fullDocument: { _id: 1, item: 'Widget', qty: 10 },
  ns: { db: 'test', coll: 'inventory' },
  documentKey: { _id: 1 }
}
```

Codebeispiele

Um ein Codebeispiel für die Verwendung der `$changeStream` Aggregationsstufe anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('inventory');

  // Open change stream
  const changeStream = collection.watch([]);

  changeStream.on('change', (change) => {
    console.log('Change detected:', change);
  });

  // Simulate insert in another operation
  setTimeout(async () => {
    await collection.insertOne({ _id: 1, item: 'Widget', qty: 10 });
  }, 1000);
```

```
  // Keep connection open to receive changes
  // In production, handle cleanup appropriately
}

example();
```

Python

```
from pymongo import MongoClient
import threading
import time

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['inventory']

    # Open change stream
    change_stream = collection.watch([])

    # Insert document in separate thread after delay
    def insert_doc():
        time.sleep(1)
        collection.insert_one({'_id': 1, 'item': 'Widget', 'qty': 10})

    threading.Thread(target=insert_doc).start()

    # Watch for changes
    for change in change_stream:
        print('Change detected:', change)
        break  # Exit after first change

    client.close()

example()
```

$ ändern StreamSplitLargeEvent

Diese Aggregationsphase wird von Elastic-Clustern nicht unterstützt.

Diese Phase ist ab Version 8.0.2 der Amazon DocumentDB-Engine verfügbar.

Die `$changeStreamSplitLargeEvent` Aggregationsstufe wird innerhalb einer $ChangeStream-Pipeline verwendet, um Ereignisse, die 16 MB überschreiten, in kleinere Fragmente aufzuteilen. Jedes Fragment wird sequentiell über den Change-Stream-Cursor zurückgegeben.

Parameter

Keine. Die Eingabe in die `$changeStreamSplitLargeEvent` Bühne sollte ein leeres Dokument sein.

Beispiel — MongoDB Shell

Das folgende Beispiel zeigt, wie die `$changeStreamSplitLargeEvent` Bühne verwendet wird, um ein übergroßes Änderungsereignis in mehrere Fragmente aufzuteilen.

Beispiel für eine Abfrage

```
// Insert a document
db.inventory.insertOne({ _id: 1, item: "Widget", payload: "x".repeat(9 * 1024 *
 1024) })

// Open change stream with the $changeStreamSplitLargeEvent stage
var changeStream = db.inventory.aggregate([
  { $changeStream: { fullDocument: "updateLookup" } },
  { $changeStreamSplitLargeEvent: {} }
]);

// Update the document
db.inventory.updateOne({ _id: 1 }, { $set: { payload: "y".repeat(9 * 1024 * 1024) } })

// Read the change event
if (changeStream.hasNext()) {
  print(tojson(changeStream.next()));
}
```

Ausgabe

```
{
  _id: { _data: '...' },
  splitEvent: { fragment: 1, of: 2},
  ns: { db: 'test', coll: 'inventory' },
  clusterTime: Timestamp(4, 1789154779),
```

```
  documentKey: { _id: 1 },
  fullDocument: { _id: 1, item: 'Widget', payload: yyyyyyyy...(9437184 chars) },
  operationType: 'update',
}
{
  _id: { _data: '...' },
  splitEvent: { fragment: 2, of: 2},
  updateDescription: { updatedFields: { payload: yyyyyyyy...(9437184 chars) },
 removedFields: [] }
}
```

Codebeispiele

Um ein Codebeispiel für die Verwendung der `$changeStreamSplitLargeEvent` Aggregationsstufe anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('inventory');

  // Insert a large document first
  await collection.insertOne({ _id: 1, item: 'Widget', payload: 'x'.repeat(9 * 1024
 * 1024) });

  // Open change stream with $changeStreamSplitLargeEvent
  const changeStream = collection.watch(
    [{ $changeStreamSplitLargeEvent: {} }],
    { fullDocument: 'updateLookup' }
  );

  changeStream.on('change', (change) => {
    console.log('Change detected:', change);
  });

  // Update to trigger a split event — fullDocument + updateDescription > 16 MB
```

```
  setTimeout(async () => {
    console.log('Triggering update...');
    await collection.updateOne({ _id: 1 }, { $set: { payload: 'y'.repeat(9 * 1024 *
 1024) } });
  }, 1000);

  // Keep connection open to receive changes
  // In production, handle cleanup appropriately
}

example();
```

Python

```
from pymongo import MongoClient
import threading
import time

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['inventory']

    # Insert large document first
    collection.drop()
    collection.insert_one({'_id': 1, 'item': 'Widget', 'payload': 'x' * (9 * 1024 *
 1024)})

    # Open change stream with $changeStreamSplitLargeEvent
    change_stream = collection.watch(
        [{'$changeStreamSplitLargeEvent': {}}],
        full_document='updateLookup'
    )

    # Update in separate thread after delay
    def update_doc():
        collection.update_one({'_id': 1}, {'$set': {'payload': 'y' * (9 * 1024 *
 1024)}})

    threading.Thread(target=update_doc).start()
```

```
    # Watch for changes
    for change in change_stream:
        print('Change detected:', change)

    client.close()

example()
```

$cmp

Der `$cmp` Operator in Amazon DocumentDB wird verwendet, um zwei Werte zu vergleichen und einen Ganzzahlwert zurückzugeben, der ihre relative Reihenfolge angibt. Es ist ein Vergleichsoperator, der zwei Ausdrücke vergleicht und einen Ganzzahlwert von -1, 0 oder 1 zurückgibt, je nachdem, ob der erste Wert kleiner, gleich oder größer als der zweite Wert ist.

Parameter

- `expression1`: Der erste Ausdruck, der verglichen werden soll.
- `expression2`: Der zweite Ausdruck, der verglichen werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung des `$cmp` Operators zum Vergleich zweier numerischer Werte.

Erstellen Sie Beispieldokumente

```
db.collection.insertMany([
  { _id: 1, value1: 10, value2: 20 },
  { _id: 2, value1: 15, value2: 15 },
  { _id: 3, value1: 20, value2: 10 }
]);
```

Beispiel für eine Abfrage

```
db.collection.find({
  $expr: {
    $cmp: ["$value1", "$value2"]
  }
})
```

Ausgabe

```
[
  { "_id" : 1, "value1" : 10, "value2" : 20 },
  { "_id" : 3, "value1" : 20, "value2" : 10 }
]
```

In diesem Beispiel vergleicht der `$cmp` Operator die `value2` Felder `value1` und für jedes Dokument. Das Ergebnis ist:

```
- `$cmp: ["$value1", "$value2"]` returns -1 for the first document (10 < 20), 0 for the
 second document (15 = 15), and 1 for the third document (20 > 10).
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$cmp` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

Hier ist ein Beispiel für die Verwendung des `$cmp` Operators in einer Node.js Anwendung mit dem `mongodb` Treiber:

```
const { MongoClient } = require('mongodb');

const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

async function main() {
  await client.connect();
  const db = client.db('test');
  const collection = db.collection('mycollection');

  // Insert sample documents
  await collection.insertMany([
    { _id: 1, value1: 10, value2: 20 },
    { _id: 2, value1: 15, value2: 15 },
    { _id: 3, value1: 20, value2: 10 }
  ]);

  // Query using $cmp operator
```

```
  const result = await collection.find({
    $expr: {
      $cmp: ['$value1', '$value2']
    }
  }).toArray();

  console.log(result);

  await client.close();
}

main();
```

Python

Hier ist ein Beispiel für die Verwendung des $cmp Operators in einer Python-Anwendung mit dem pymongo Treiber:

```
from pymongo import MongoClient

client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
db = client['test']
collection = db['mycollection']

# Insert sample documents
collection.insert_many([
    {'_id': 1, 'value1': 10, 'value2': 20},
    {'_id': 2, 'value1': 15, 'value2': 15},
    {'_id': 3, 'value1': 20, 'value2': 10}
])

# Query using $cmp operator
result = list(collection.find({
    '$expr': {
        '$cmp': ['$value1', '$value2']
    }
}))

print(result)

client.close()
```

Die Ausgabe sowohl des als auch des Node.js Python-Beispiels entspricht der des MongoDB-Shell-Beispiels:

```
[
  { "_id" : 1, "value1" : 10, "value2" : 20 },
  { "_id" : 2, "value1" : 15, "value2" : 15 },
  { "_id" : 3, "value1" : 20, "value2" : 10 }
]
```

$collStats

Neu ab Version 4.0

Die `$collStats` Aggregationsphase in Amazon DocumentDB bietet Statistiken über die angegebene Sammlung, ähnlich dem `db.collection.stats()` Befehl in der MongoDB-Shell. In dieser Phase können Informationen über die Sammlung abgerufen werden, z. B. die Anzahl der Dokumente, die Gesamtgröße der Sammlung und verschiedene Leistungskennzahlen.

Parameter

- `latencyStats`: (optional) Ein Dokument, das die Optionen für die Erfassung von Latenzstatistiken spezifiziert. Dieser Parameter wird in Amazon DocumentDB nicht unterstützt.
- `recordStats`: (optional) Ein Dokument, das die Optionen für die Erfassung von Datensatzstatistiken festlegt. Dieser Parameter wird in Amazon DocumentDB nicht unterstützt.
- `queryExecStats`: (optional) Ein Dokument, das die Optionen für die Erfassung von Statistiken zur Abfrageausführung spezifiziert. Dieser Parameter wird in Amazon DocumentDB nicht unterstützt.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie die `$collStats` Aggregationsphase verwendet wird, um Statistiken über eine `test` in der db Datenbank benannte Sammlung abzurufen.

Erstellen Sie Beispieldokumente

```
db.test.insertMany([
  { "name": "John", "age": 30 },
  { "name": "Jane", "age": 25 },
```

```
  { "name": "Bob", "age": 35 }
]);
```

Beispiel für eine Abfrage

```
db.test.aggregate([
  { $collStats: {} }
]);
```

Ausgabe

```
{
  "ns" : "db.test",
  "count" : 3,
  "size" : 87,
  "avgObjSize" : 29,
  "storageSize" : 40960,
  "capped" : false,
  "nindexes" : 1,
  "totalIndexSize" : 8192,
  "indexSizes" : {
    "_id_" : 8192
  },
  "collScans" : 0,
  "idxScans" : 0,
  "opCounter" : {
    "numDocsIns" : 3,
    "numDocsUpd" : 0,
    "numDocsDel" : 0
  },
  "cacheStats" : {
    "collBlksHit" : 0,
    "collBlksRead" : 0,
    "collHitRatio" : 0,
    "idxBlksHit" : 0,
    "idxBlksRead" : 0,
    "idxHitRatio" : 0
  },
  "lastReset" : "2023-04-11T12:00:00Z",
  "ok" : 1,
  "operationTime" : Timestamp(1681206000, 1)
}
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$collStats` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

Hier ist ein Beispiel für die Verwendung der `$collStats` Aggregationsphase in einer Node.js Anwendung, die den offiziellen Node.js MongoDB-Treiber verwendet:

```
const { MongoClient } = require('mongodb');

async function runCollStatsExample() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('db');
  const collection = db.collection('test');

  const result = await collection.aggregate([
    { $collStats: {} }
  ]).toArray();

  console.log(result);

  await client.close();
}

runCollStatsExample();
```

Python

Hier ist ein Beispiel für die Verwendung der `$collStats` Aggregationsphase in einer Python-Anwendung mithilfe des PyMongo Treibers:

```
from pymongo import MongoClient

def run_coll_stats_example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['db']
    collection = db['test']
```

```
    result = list(collection.aggregate([
        { '$collStats': {} }
    ]))

    print(result)

    client.close()

run_coll_stats_example()
```

$concat

Der `$concat` Aggregationsoperator in Amazon DocumentDB verkettet (oder kombiniert) mehrere Zeichenketten in einem Dokument, um eine einzige Zeichenfolge zu erzeugen, die an die Anwendung zurückgegeben werden kann. Dies reduziert den Arbeitsaufwand in der Anwendung, da die Bearbeitung der Zeichenfolge auf Datenbankebene erfolgt.

Parameter

- `expression1`: Die erste Zeichenfolge, die verkettet werden soll.
- `expression2`: Die zweite Zeichenfolge, die verkettet werden soll.
- `...`: Zusätzliche Zeichenketten zum Verketten (optional).

Beispiel (MongoDB Shell)

In diesem Beispiel verketten wir die Vor- und Nachnamen der Benutzer, um den vollständigen Namen jeder Person zu erhalten.

Erstellen Sie Beispieldokumente

```
db.people.insertMany([
  { "_id":1, "first_name":"Jane", "last_name":"Doe", "DOB":"2/1/1999", "Desk": "MSP102-
MN"},
  { "_id":2, "first_name":"John", "last_name":"Doe", "DOB":"12/21/1992", "Desk":
 "DSM301-IA"},
  { "_id":3, "first_name":"Steve", "last_name":"Smith", "DOB":"3/21/1981",
 "Desk":"MKE233-WI"}
])
```

Beispiel für eine Abfrage

```
db.people.aggregate([
  { $project: { full_name: { $concat: [ "$first_name", " ", "$last_name"] } } }
])
```

Ausgabe

```
{ "_id" : 1, "full_name" : "Jane Doe" }
{ "_id" : 2, "full_name" : "John Doe" }
{ "_id" : 3, "full_name" : "Steve Smith" }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$concat` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function concatenateNames() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');

  const result = await db.collection('people').aggregate([
    { $project: { full_name: { $concat: [ "$first_name", " ", "$last_name"] } } }
  ]).toArray();

  console.log(result);

  await client.close();
}

concatenateNames();
```

Python

```
from pymongo import MongoClient
```

```
def concatenate_names():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']

    result = list(db.people.aggregate([
        { '$project': { 'full_name': { '$concat': [ '$first_name', ' ',
 '$last_name' ] } } }
    ]))

    print(result)

    client.close()

concatenate_names()
```

$concatArrays

Der `$concatArrays` Aggregationsoperator in Amazon DocumentDB wird verwendet, um zwei oder mehr Arrays zu einem einzigen Array zu verketten. Dies kann nützlich sein, wenn Sie mehrere Datenarrays zur weiteren Verarbeitung oder Analyse zu einem einzigen Array kombinieren müssen.

Parameter

- `array1`: Das erste Array, das verkettet werden soll.
- `array2`: Das zweite Array, das verkettet werden soll.
- `[array3, ...]`: (optional) Zusätzliche Arrays, die verkettet werden sollen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$concatArrays` Operator verwendet wird, um zwei Arrays zu einem einzigen Array zu kombinieren.

Erstellen Sie Beispieldokumente

```
db.collection.insertMany([
  {
    "_id": 1,
    "name": "John Doe",
    "hobbies": ["reading", "swimming"],
```

```
    "skills": ["programming", "design"]
  },
  {
    "_id": 2,
    "name": "Jane Smith",
    "hobbies": ["hiking", "cooking"],
    "skills": ["marketing", "analysis"]
  }
]);
```

Beispiel für eine Abfrage

```
db.collection.aggregate([
  {
    $project: {
      _id: 0,
      name: 1,
      all_activities: { $concatArrays: ["$hobbies", "$skills"] }
    }
  }
]);
```

Ausgabe

```
[
  {
    "name": "John Doe",
    "all_activities": [
      "reading",
      "swimming",
      "programming",
      "design"
    ]
  },
  {
    "name": "Jane Smith",
    "all_activities": [
      "hiking",
      "cooking",
      "marketing",
      "analysis"
    ]
  }
```

```
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$concatArrays` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('mydb');
  const collection = db.collection('mycollection');

  const result = await collection.aggregate([
    {
      $project: {
        _id: 0,
        name: 1,
        all_activities: { $concatArrays: ['$hobbies', '$skills'] }
      }
    }
  ]).toArray();

  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['mydb']
```

```
    collection = db['mycollection']

    result = list(collection.aggregate([
        {
            '$project': {
                '_id': 0,
                'name': 1,
                'all_activities': { '$concatArrays': ['$hobbies', '$skills'] }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$cond

Der `$cond` Operator in Amazon DocumentDB wird verwendet, um einen bedingten Ausdruck auszuwerten und einen von zwei möglichen Ergebnisausdrücken zurückzugeben.

Parameter

- `if`: Der boolesche Ausdruck, der ausgewertet werden soll.
- `then`: Der Ausdruck, der zurückgegeben werden soll, wenn der `if` Ausdruck wahr ist.
- `else`: Der Ausdruck, der zurückgegeben werden soll, wenn der `if` Ausdruck falsch ist.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$cond` Operator verwendet wird, um einen Wert zurückzugeben, der auf dem Alter einer Person basiert.

Erstellen Sie Beispieldokumente

```
db.people.insertMany([
  { _id: 1, name: "John Doe", age: 35 },
  { _id: 2, name: "Jane Doe", age: 25 },
  { _id: 3, name: "Bob Smith", age: 65 }
]);
```

Beispiel für eine Abfrage

```
db.people.aggregate([
  {
    $project: {
      name: 1,
      ageGroup: {
        $cond: {
          if: { $lt: ["$age", 30] },
          then: "young",
          else: {
            $cond: {
              if: { $lt: ["$age", 65] },
              then: "middle-aged",
              else: "elderly"
            }
          }
        }
      }
    }
  }
])
```

Ausgabe

```
[
  { "_id" : 1, "name" : "John Doe", "ageGroup" : "middle-aged" },
  { "_id" : 2, "name" : "Jane Doe", "ageGroup" : "young" },
  { "_id" : 3, "name" : "Bob Smith", "ageGroup" : "elderly" }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$cond` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
```

```
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('people');

  const result = await collection.aggregate([
    { $project: {
        name: 1,
        ageGroup: {
            $cond: {
                if: { $lt: ["$age", 30] },
                then: "young",
                else: {
                    $cond: {
                        if: { $lt: ["$age", 65] },
                        then: "middle-aged",
                        else: "elderly"
                    }
                }
            }
        }
      }
    }
  ]).toArray();

  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient


def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.people
```

```
    result = list(collection.aggregate([
        {
            '$project': {
                'name': 1,
                'ageGroup': {
                    '$cond': {
                        'if': { '$lt': ["$age", 30]},
                        'then': "young",
                        'else': {
                            '$cond': {
                                'if': { '$lt': ["$age", 65]},
                                'then': "middle-aged",
                                'else': "elderly"
                            }
                        }
                    }
                }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$ konvertieren

Neu ab Version 4.0

Der `$convert` Operator in Amazon DocumentDB wird verwendet, um einen Wert von einem Datentyp in einen anderen zu konvertieren. Dieser Operator ist nützlich, wenn Sie Operationen mit Daten verschiedener Typen ausführen müssen, z. B. die Konvertierung einer Zeichenfolge in eine Zahl oder eines Datums in einen Zeitstempel.

Parameter

- `to`: Der Zieldatentyp, in den der Wert konvertiert werden soll. Unterstützte Werte sind `"string"`, `"double"`, `"long"`, `"int"`, `"date"` und `"boolean"`.

- `from`: Der aktuelle Datentyp des Werts. Falls nicht angegeben, versucht Amazon DocumentDB, den Datentyp automatisch zu erkennen.
- `onError`: (optional) Der Wert, der zurückgegeben werden soll, wenn die Konvertierung fehlschlägt. Kann ein bestimmter Wert oder einer der folgenden Sonderwerte sein:`"null"`,`"zerofill"`, oder`"error"`.
- `onNull`: (optional) Der Wert, der zurückgegeben werden soll, wenn der Eingabewert`null`. Kann ein bestimmter Wert oder einer der folgenden Sonderwerte sein:`"null"`,`"zerofill"`, oder`"error"`.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Konvertierung eines Zeichenkettenwerts in ein Datum mithilfe des `$convert` Operators.

Erstellen Sie Beispieldokumente

```
db.users.insertMany([
  { _id: 1, name: "John Doe", joinedOn: "2022-01-01" },
  { _id: 2, name: "Jane Smith", joinedOn: "2023-02-15" },
  { _id: 3, name: "Bob Johnson", joinedOn: "invalid date" }
]);
```

Beispiel für eine Abfrage

```
db.users.aggregate([
  {
    $project: {
      _id: 1,
      name: 1,
      joinedOn: {
        $convert: {
          input: "$joinedOn",
          to: "date",
          onError: "null",
          onNull: "null"
        }
      }
    }
  }
])
```

Ausgabe

```
[
  { "_id" : 1, "name" : "John Doe", "joinedOn" : ISODate("2022-01-01T00:00:00Z") },
  { "_id" : 2, "name" : "Jane Smith", "joinedOn" : ISODate("2023-02-15T00:00:00Z") },
  { "_id" : 3, "name" : "Bob Johnson", "joinedOn" : null }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$convert` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require("mongodb");

async function example() {
  const client = await MongoClient.connect("mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false");
  const db = client.db("test");
  const users = db.collection("users");

  const results = await users.aggregate([
    {
      $project: {
        _id: 1,
        name: 1,
        joinedOn: {
          $convert: {
            input: "$joinedOn",
            to: "date",
            onError: "null",
            onNull: "null"
          }
        }
      }
    }
  ]).toArray();

  console.log(results);
  client.close();
```

```
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient("mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false")
    db = client["test"]
    users = db["users"]

    results = list(users.aggregate([
        {
            "$project": {
                "_id": 1,
                "name": 1,
                "joinedOn": {
                    "$convert": {
                        "input": "$joinedOn",
                        "to": "date",
                        "onError": "null",
                        "onNull": "null"
                    }
                }
            }
        }
    ]))

    print(results)
    client.close()

example()
```

$cos

Neu ab Version 8.0.1.

Der `$cos` Operator in Amazon DocumentDB gibt den Kosinus eines Werts zurück, der im Bogenmaß gemessen wird. Verwenden Sie ihn in der Aggregationspipeline, um trigonometrische Berechnungen für numerische Felder durchzuführen.

Parameter

- `expression`: Ein Ausdruck, der in eine Zahl im Bogenmaß aufgelöst wird.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$cos` Operator verwendet wird, um den Kosinus von Winkelwerten im Bogenmaß zu berechnen.

Erstellen Sie Beispieldokumente

```
db.angles.insertMany([
  { "_id": 1, "angle": 0 },
  { "_id": 2, "angle": 1.0471975511965976 },
  { "_id": 3, "angle": 3.141592653589793 }
]);
```

Beispiel für eine Abfrage

```
db.angles.aggregate([
  { $project: {
    "cosine": { $cos: "$angle" }
  }}
]);
```

Ausgabe

```
[
  { "_id": 1, "cosine": 1 },
  { "_id": 2, "cosine": 0.5000000000000001 },
  { "_id": 3, "cosine": -1 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$cos` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('angles');

  const result = await collection.aggregate([
    { $project: {
      "cosine": { $cos: "$angle" }
    }}
  ]).toArray();

  console.log(result);
  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['angles']

    result = list(collection.aggregate([
        { '$project': {
            'cosine': { '$cos': '$angle' }
        }}
    ]))

    print(result)
    client.close()
```

```
if __name__ == "__main__":
    main()
```

$cosh

Neu ab Version 8.0.1.

Der `$cosh` Operator in Amazon DocumentDB gibt den hyperbolischen Kosinus eines Werts zurück. Verwenden Sie ihn in der Aggregationspipeline, um hyperbolische trigonometrische Berechnungen für numerische Felder durchzuführen.

Parameter

- `expression`: Ein Ausdruck, der in eine Zahl aufgelöst wird.

Der Rückgabetyp ist `double` standardmäßig. Wenn die Eingabe eine 128-Bit-Dezimalzahl ist, ist die Ausgabe ebenfalls eine 128-Bit-Dezimalzahl.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$cosh` Operator verwendet wird, um den hyperbolischen Kosinus numerischer Werte zu berechnen.

Erstellen Sie Beispieldokumente

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 1 },
  { "_id": 3, "value": -1 }
]);
```

Beispiel für eine Abfrage

```
db.values.aggregate([
  { $project: {
    "result": { $cosh: "$value" }
  }}
]);
```

Ausgabe

```
[
  { "_id": 1, "result": 1 },
  { "_id": 2, "result": 1.5430806348152437 },
  { "_id": 3, "result": 1.5430806348152437 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$cosh` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('values');

  const result = await collection.aggregate([
    { $project: {
      "result": { $cosh: "$value" }
    }}
  ]).toArray();

  console.log(result);
  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
```

```
    db = client['test']
    collection = db['values']

    result = list(collection.aggregate([
        { '$project': {
            'result': { '$cosh': '$value' }
        }}
    ]))

    print(result)
    client.close()

if __name__ == "__main__":
    main()
```

$count (Stufe)

Die `$count` Aggregationsstufe in Amazon DocumentDB wird verwendet, um die Anzahl der Dokumente zu zählen, die in die Phase übergehen. Sie wird häufig als letzte Phase in einer Aggregationspipeline verwendet, um die Gesamtzahl der Dokumente zurückzugeben, die den vorherigen Phasen entsprechen.

Parameter

- `field`: Das zu zählende Feld. Dieser Parameter ist optional. Wenn er nicht angegeben wird, zählt der Schritt die Gesamtzahl der Eingabedokumente.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie die `$count` Bühne verwendet wird, um die Gesamtzahl der Dokumente in einer Sammlung abzurufen.

Erstellen Sie Beispieldokumente

```
db.users.insertMany([
  { name: "John", age: 30 },
  { name: "Jane", age: 25 },
  { name: "Bob", age: 35 },
  { name: "Alice", age: 28 }
]);
```

Beispiel für eine Abfrage

```
db.users.aggregate([
  { $count: "total" }
]);
```

Ausgabe

```
{ "total" : 4 }
```

Das Beispiel aggregiert die `users` Sammlung und verwendet die `$count` Stufe, um die Gesamtzahl der Dokumente zu zählen.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$count` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function countDocuments() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('mydb');
  const collection = db.collection('users');

  const result = await collection.aggregate([
    { $count: "total" }
  ]).toArray();

  console.log(result[0].total);

  await client.close();
}

countDocuments();
```

Python

```
from pymongo import MongoClient

def count_documents():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['mydb']
    collection = db['users']

    result = list(collection.aggregate([
        { '$count': "total" }
    ]))

    print(result[0]['total'])

    client.close()

count_documents()
```

$count (Akkumulator)

Neu ab Version 8.0.1.

Verwenden Sie den `$count` Akkumulator innerhalb einer `$group` Stufe, um die Anzahl der Dokumente in jeder Gruppe zurückzugeben. Es akzeptiert ein leeres Objekt `{}` als Argument.

Dies unterscheidet sich von der `$count` Pipeline-Phase, bei der es sich um eine eigenständige Phase handelt, in der alle Dokumente gezählt werden, die die Pipeline durchlaufen. Der `$count` Akkumulator zählt stattdessen die Dokumente innerhalb jeder Gruppe, die von `$group` erstellt wurde.

Syntax

```
{ $count: {} }
```

Parameter

- Der `$count` Akkumulator akzeptiert keine Argumente. Er akzeptiert ein leeres Objekt `{}` und gibt die Anzahl der Dokumente in jeder Gruppe zurück.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$count` Akkumulator verwendet wird, um die Anzahl der Produkte in jeder Kategorie zu zählen.

Erstellen Sie Beispieldokumente

```
db.products.insertMany([
  { name: "Widget", category: "A" },
  { name: "Gadget", category: "A" },
  { name: "Doohickey", category: "B" },
  { name: "Thingamajig", category: "B" },
  { name: "Whatsit", category: "B" }
])
```

Beispiel für eine Abfrage

```
db.products.aggregate([
  { $group: { _id: "$category", count: { $count: {} } } }
])
```

Ausgabe

```
[
  { "_id": "A", "count": 2 },
  { "_id": "B", "count": 3 }
]
```

Beispiel für die Verwendung von Fensteroperatoren (MongoDB Shell)

Neu ab Version 8.0.2.

Der `$count` Akkumulator kann auch als Fensterantrieb auf der `$setWindowFields` Bühne verwendet werden. In diesem Zusammenhang gibt es die Anzahl der Dokumente in jedem Fenster zurück. Sie geben den Operator unter dem `output` Feld an und definieren optional die Fenstergrenzen mit einem `window` Dokument.

Erstellen Sie Beispieldokumente

```
db.events.insertMany([
```

```
  { _id: 1, category: "A", day: 1 },
  { _id: 2, category: "A", day: 2 },
  { _id: 3, category: "A", day: 3 },
  { _id: 4, category: "B", day: 1 },
  { _id: 5, category: "B", day: 2 }
])
```

Beispiel für eine Abfrage

Das folgende Beispiel partitioniert die Dokumente nach`category`, sortiert jede Partition nach `day` und gibt eine fortlaufende Anzahl von Dokumenten vom Anfang der Partition bis zum aktuellen Dokument zurück.

```
db.events.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$category",
      sortBy: { day: 1 },
      output: {
        runningCount: {
          $count: {},
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
])
```

Ausgabe

```
[
  { "_id": 1, "category": "A", "day": 1, "runningCount": 1 },
  { "_id": 2, "category": "A", "day": 2, "runningCount": 2 },
  { "_id": 3, "category": "A", "day": 3, "runningCount": 3 },
  { "_id": 4, "category": "B", "day": 1, "runningCount": 1 },
  { "_id": 5, "category": "B", "day": 2, "runningCount": 2 }
]
```

Jedes Dokument wird um die Anzahl der Dokumente in seiner Partition bis einschließlich des aktuellen Dokuments erweitert. `runningCount`

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$count` Akkumulators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten. Die folgenden Beispiele zeigen sowohl die Verwendung des Akkumulators (in`$group`) als auch die Verwendung von Fensteroperatoren (in`$setWindowFields`):

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const uri = 'mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false';
  const client = new MongoClient(uri);

  try {
    await client.connect();

    const db = client.db('test');

    // Accumulator usage: count per group
    const products = db.collection('products');
    await products.insertMany([
      { name: "Widget", category: "A" },
      { name: "Gadget", category: "A" },
      { name: "Doohickey", category: "B" },
      { name: "Thingamajig", category: "B" },
      { name: "Whatsit", category: "B" }
    ]);
    const accumulatorResult = await products.aggregate([
      { $group: { _id: "$category", count: { $count: {} } } }
    ]).toArray();
    console.log('Accumulator result:', accumulatorResult);

    // Window operator usage: running count within each partition
    const events = db.collection('events');
    await events.insertMany([
      { _id: 1, category: "A", day: 1 },
      { _id: 2, category: "A", day: 2 },
      { _id: 3, category: "A", day: 3 },
      { _id: 4, category: "B", day: 1 },
```

```
      { _id: 5, category: "B", day: 2 }
    ]);
    const windowResult = await events.aggregate([
      {
        $setWindowFields: {
          partitionBy: "$category",
          sortBy: { day: 1 },
          output: {
            runningCount: {
              $count: {},
              window: { documents: ["unbounded", "current"] }
            }
          }
        }
      }
    ]).toArray();
    console.log('Window result:', windowResult);

  } catch (error) {
    console.error('Error:', error);
  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient
from pprint import pprint

def example():
    client = None
    try:
        client = MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

        db = client['test']

        # Accumulator usage: count per group
        products = db['products']
```

```
        products.insert_many([
            { 'name': 'Widget', 'category': 'A' },
            { 'name': 'Gadget', 'category': 'A' },
            { 'name': 'Doohickey', 'category': 'B' },
            { 'name': 'Thingamajig', 'category': 'B' },
            { 'name': 'Whatsit', 'category': 'B' }
        ])
        accumulator_result = list(products.aggregate([
            { '$group': { '_id': '$category', 'count': { '$count': {} } } }
        ]))
        pprint(accumulator_result)

        # Window operator usage: running count within each partition
        events = db['events']
        events.insert_many([
            { '_id': 1, 'category': 'A', 'day': 1 },
            { '_id': 2, 'category': 'A', 'day': 2 },
            { '_id': 3, 'category': 'A', 'day': 3 },
            { '_id': 4, 'category': 'B', 'day': 1 },
            { '_id': 5, 'category': 'B', 'day': 2 }
        ])
        window_result = list(events.aggregate([
            {
                '$setWindowFields': {
                    'partitionBy': '$category',
                    'sortBy': { 'day': 1 },
                    'output': {
                        'runningCount': {
                            '$count': {},
                            'window': { 'documents': ['unbounded', 'current'] }
                        }
                    }
                }
            }
        ]))
        pprint(window_result)

    except Exception as e:
        print(f"An error occurred: {e}")

    finally:
        if client:
            client.close()
```

```
example()
```

$ CovariancePop

Neu ab Version 8.0.2.

Der `$covariancePop` Operator in Amazon DocumentDB gibt die Populationskovarianz zweier numerischer Ausdrücke zurück. Es handelt sich um einen Fensteroperator, der nur in der `$setWindowFields` Phase verwendet wird. In den `$group` Stufen oder ist er nicht gültig. `$bucket` Sie geben den Operator unter dem `output` Feld der Bühne an und definieren optional die Fenstergrenzen mit einem `window` Dokument.

Parameter

- `$covariancePop`verwendet ein Array mit zwei Elementen`[ <expression1>, <expression2> ]`, wobei jedes Element ein numerischer Ausdruck ist (ein Feldpfad`"$x"`, z. B. ein Ausdruck oder eine Konstante). Es gibt die Populationskovarianz der beiden Ausdrücke in den Dokumenten im Fenster zurück.

Beispiel (MongoDB Shell)

Das folgende Beispiel partitioniert die Dokumente nach`series`, sortiert jede Partition nach `t` und berechnet die Populationskovarianz von `x` und `y` über die gesamte Partition.

Erstellen Sie Beispieldokumente

```
db.measurements.insertMany([
  { _id: 1, series: "A", t: 1, x: 1, y: 2 },
  { _id: 2, series: "A", t: 2, x: 2, y: 5 },
  { _id: 3, series: "A", t: 3, x: 3, y: 8 },
  { _id: 4, series: "B", t: 1, x: 1, y: 8 },
  { _id: 5, series: "B", t: 2, x: 2, y: 5 },
  { _id: 6, series: "B", t: 3, x: 3, y: 2 }
]);
```

Beispiel für eine Abfrage

```
db.measurements.aggregate([
  {
```

```
    $setWindowFields: {
      partitionBy: "$series",
      sortBy: { t: 1 },
      output: {
        covariance: {
          $covariancePop: [ "$x", "$y" ],
          window: { documents: ["unbounded", "unbounded"] }
        }
      }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "series": "A", "t": 1, "x": 1, "y": 2, "covariance": 2 },
  { "_id": 2, "series": "A", "t": 2, "x": 2, "y": 5, "covariance": 2 },
  { "_id": 3, "series": "A", "t": 3, "x": 3, "y": 8, "covariance": 2 },
  { "_id": 4, "series": "B", "t": 1, "x": 1, "y": 8, "covariance": -2 },
  { "_id": 5, "series": "B", "t": 2, "x": 2, "y": 5, "covariance": -2 },
  { "_id": 6, "series": "B", "t": 3, "x": 3, "y": 2, "covariance": -2 }
]
```

Jedes Dokument in einer Partition erhält dieselbe `covariance` Kovarianz der Grundgesamtheit aller Dokumente in dieser Partition `x` und `y` für alle Dokumente in dieser Partition: `2` für Reihen `A` (X und Y nehmen zusammen zu) und `-2` für Reihen `B` (y nimmt ab, wenn X zunimmt).

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$covariancePop` Operators in der `$setWindowFields` Phase anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
```

```
  try {
    await client.connect();
    const db = client.db('test');
    const measurements = db.collection('measurements');

    await measurements.insertMany([
      { _id: 1, series: "A", t: 1, x: 1, y: 2 },
      { _id: 2, series: "A", t: 2, x: 2, y: 5 },
      { _id: 3, series: "A", t: 3, x: 3, y: 8 },
      { _id: 4, series: "B", t: 1, x: 1, y: 8 },
      { _id: 5, series: "B", t: 2, x: 2, y: 5 },
      { _id: 6, series: "B", t: 3, x: 3, y: 2 }
    ]);

    const result = await measurements.aggregate([
      {
        $setWindowFields: {
          partitionBy: "$series",
          sortBy: { t: 1 },
          output: {
            covariance: {
              $covariancePop: [ "$x", "$y" ],
              window: { documents: ["unbounded", "unbounded"] }
            }
          }
        }
      }
    ]).toArray();

    console.log(result);
  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient

def example():
```

```
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    try:
        db = client['test']
        measurements = db['measurements']

        measurements.insert_many([
            { '_id': 1, 'series': 'A', 't': 1, 'x': 1, 'y': 2 },
            { '_id': 2, 'series': 'A', 't': 2, 'x': 2, 'y': 5 },
            { '_id': 3, 'series': 'A', 't': 3, 'x': 3, 'y': 8 },
            { '_id': 4, 'series': 'B', 't': 1, 'x': 1, 'y': 8 },
            { '_id': 5, 'series': 'B', 't': 2, 'x': 2, 'y': 5 },
            { '_id': 6, 'series': 'B', 't': 3, 'x': 3, 'y': 2 }
        ])

        result = list(measurements.aggregate([
            {
                '$setWindowFields': {
                    'partitionBy': '$series',
                    'sortBy': { 't': 1 },
                    'output': {
                        'covariance': {
                            '$covariancePop': [ '$x', '$y' ],
                            'window': { 'documents': ['unbounded', 'unbounded'] }
                        }
                    }
                }
            }
        ]))

        print(result)
    finally:
        client.close()

example()
```

$KovarianzSAMP

Neu ab Version 8.0.2.

Der `$covarianceSamp` Operator in Amazon DocumentDB gibt die Stichprobenkovarianz zweier numerischer Ausdrücke zurück. Es handelt sich um einen Fensteroperator, der nur in der `$setWindowFields` Phase verwendet wird. In den `$group` Stufen oder ist er nicht gültig. `$bucket` Sie geben den Operator unter dem `output` Feld der Bühne an und definieren optional die Fenstergrenzen mit einem `window` Dokument.

Parameter

- `$covarianceSamp`verwendet ein Array mit zwei Elementen`[ <expression1>, <expression2> ]`, wobei jedes Element ein numerischer Ausdruck ist (ein Feldpfad`"$x"`, z. B. ein Ausdruck oder eine Konstante). Es gibt die Beispielkovarianz der beiden Ausdrücke in den Dokumenten im Fenster zurück.

Beispiel (MongoDB Shell)

Das folgende Beispiel partitioniert die Dokumente nach`series`, sortiert jede Partition nach `t` und berechnet die Stichprobenkovarianz von `x` und `y` über die gesamte Partition.

Erstellen Sie Beispieldokumente

```
db.measurements.insertMany([
  { _id: 1, series: "A", t: 1, x: 1, y: 2 },
  { _id: 2, series: "A", t: 2, x: 2, y: 5 },
  { _id: 3, series: "A", t: 3, x: 3, y: 8 },
  { _id: 4, series: "B", t: 1, x: 1, y: 8 },
  { _id: 5, series: "B", t: 2, x: 2, y: 5 },
  { _id: 6, series: "B", t: 3, x: 3, y: 2 }
]);
```

Beispiel für eine Abfrage

```
db.measurements.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$series",
      sortBy: { t: 1 },
      output: {
        covariance: {
          $covarianceSamp: [ "$x", "$y" ],
          window: { documents: ["unbounded", "unbounded"] }
        }
```

```
      }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "series": "A", "t": 1, "x": 1, "y": 2, "covariance": 3 },
  { "_id": 2, "series": "A", "t": 2, "x": 2, "y": 5, "covariance": 3 },
  { "_id": 3, "series": "A", "t": 3, "x": 3, "y": 8, "covariance": 3 },
  { "_id": 4, "series": "B", "t": 1, "x": 1, "y": 8, "covariance": -3 },
  { "_id": 5, "series": "B", "t": 2, "x": 2, "y": 5, "covariance": -3 },
  { "_id": 6, "series": "B", "t": 3, "x": 3, "y": 2, "covariance": -3 }
]
```

Jedes Dokument in einer Partition erhält dieselbe `covariance` Stichprobenkovarianz aller Dokumente in dieser Partition `x` und `y` für alle Dokumente in dieser Partition: `3` für Reihen `A` (X und Y nehmen zusammen zu) und `-3` für Reihen `B` (y nimmt ab, wenn X zunimmt). Die Stichprobenkovarianz wird durch `n - 1` statt durch als dividiertn, sodass ihr Betrag größer ist als die Kovarianz der Grundgesamtheit für dieselben Daten.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$covarianceSamp` Operators in der `$setWindowFields` Phase anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  try {
    await client.connect();
    const db = client.db('test');
    const measurements = db.collection('measurements');
```

```
    await measurements.insertMany([
      { _id: 1, series: "A", t: 1, x: 1, y: 2 },
      { _id: 2, series: "A", t: 2, x: 2, y: 5 },
      { _id: 3, series: "A", t: 3, x: 3, y: 8 },
      { _id: 4, series: "B", t: 1, x: 1, y: 8 },
      { _id: 5, series: "B", t: 2, x: 2, y: 5 },
      { _id: 6, series: "B", t: 3, x: 3, y: 2 }
    ]);

    const result = await measurements.aggregate([
      {
        $setWindowFields: {
          partitionBy: "$series",
          sortBy: { t: 1 },
          output: {
            covariance: {
              $covarianceSamp: [ "$x", "$y" ],
              window: { documents: ["unbounded", "unbounded"] }
            }
          }
        }
      }
    ]).toArray();

    console.log(result);
  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    try:
        db = client['test']
```

```
        measurements = db['measurements']

        measurements.insert_many([
            { '_id': 1, 'series': 'A', 't': 1, 'x': 1, 'y': 2 },
            { '_id': 2, 'series': 'A', 't': 2, 'x': 2, 'y': 5 },
            { '_id': 3, 'series': 'A', 't': 3, 'x': 3, 'y': 8 },
            { '_id': 4, 'series': 'B', 't': 1, 'x': 1, 'y': 8 },
            { '_id': 5, 'series': 'B', 't': 2, 'x': 2, 'y': 5 },
            { '_id': 6, 'series': 'B', 't': 3, 'x': 3, 'y': 2 }
        ])

        result = list(measurements.aggregate([
            {
                '$setWindowFields': {
                    'partitionBy': '$series',
                    'sortBy': { 't': 1 },
                    'output': {
                        'covariance': {
                            '$covarianceSamp': [ '$x', '$y' ],
                            'window': { 'documents': ['unbounded', 'unbounded'] }
                        }
                    }
                }
            }
        ]))

        print(result)
    finally:
        client.close()

example()
```

$currentOp

Die `$currentOp` Aggregationsphase gibt Informationen über aktuell ausgeführte Operationen in der Datenbank zurück. Diese Phase ist nützlich für die Überwachung aktiver Abfragen und Operationen in einer Aggregationspipeline.

Parameter

- `allUsers`(optional): Wenn auf gesetzt`true`, werden Operationen für alle Benutzer zurückgegeben. Der Standardwert ist `false`.

- `idleConnections`(optional): Wenn auf gesetzt`true`, schließt dies inaktive Verbindungen ein. Der Standardwert ist `false`.
- `idleCursors`(optional): Wenn auf gesetzt`true`, enthält dies Informationen über Cursor im Leerlauf. Der Standardwert ist `false`.
- `idleSessions`(optional): Wenn auf gesetzt`true`, enthält es Informationen über inaktive Sitzungen. Der Standardwert ist `true`.
- `localOps`(optional): Wenn auf gesetzt`true`, schließt dies lokale Operationen ein. Der Standardwert ist `false`.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie die `$currentOp` Aggregationsphase verwendet wird, um Informationen über aktive Lesevorgänge abzurufen.

Beispiel für eine Abfrage

```
db.aggregate([
  { $currentOp: { allUsers: true, idleConnections: false } },
  { $match: { op: "query" } }
])
```

Ausgabe

```
[
  {
    "opid": "12345",
    "active": true,
    "op": "query",
    "ns": "test.users",
    "secs_running": 2
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$currentOp` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('admin');

  const result = await db.aggregate([
    { $currentOp: { allUsers: true, idleConnections: false } },
    { $match: { op: "query" } }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['admin']

    result = list(db.aggregate([
        { '$currentOp': { 'allUsers': True, 'idleConnections': False } },
        { '$match': { 'op': 'query' } }
    ]))

    print(result)
    client.close()

example()
```

$DateAdd

Neu ab Version 5.0

Mit dem `$dateAdd` Aggregationsoperator in Amazon DocumentDB können Sie einem Datums- und Uhrzeitwert eine Dauer hinzufügen.

Parameter

- `date`: Ein Datums- und Uhrzeitwert, dem eine Dauer hinzugefügt werden soll.
- `duration`: Die Dauer, die dem `date` Wert hinzugefügt werden soll. Dies kann als Objekt mit Schlüsseln für`years`,,`months`, `weeks` `days` `hours``minutes`, und angegeben `seconds` werden.
- `timezone`: (optional) Die Zeitzone, die beim Hinzufügen des Datums verwendet werden soll. Falls nicht angegeben, wird die Standardzeitzone des Amazon DocumentDB-Clusters verwendet.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$dateAdd` Operator verwendet wird, um einem Datum 2 Tage und 12 Stunden hinzuzufügen.

Erstellen Sie Beispieldokumente

```
db.events.insertMany([
  { _id: 1, eventDate: ISODate("2023-04-01T10:00:00Z") },
  { _id: 2, eventDate: ISODate("2023-04-02T12:00:00Z") },
  { _id: 3, eventDate: ISODate("2023-04-03T14:00:00Z") }
]);
```

Beispiel für eine Abfrage

```
db.events.aggregate([
  {
    $project: {
      _id: 1,
      eventDate: 1,
      eventDatePlustwodaysandtwelvehours: {
        $dateAdd: {
          startDate: {
            $dateAdd: {
              startDate: "$eventDate",
```

```
              unit: "day",
              amount: 2
            }
          },
          unit: "hour",
          amount: 12
        }
      }
    }
  }
])
```

Ausgabe

```
[
  {
    "_id": 1,
    "eventDate": "2023-04-01T10:00:00Z",
    "eventDatePlustwodaysandtwelvehours": ISODate("2023-04-03T22:00:00Z)"
  },
  {
    "_id": 2,
    "eventDate": "2023-04-02T12:00:00Z",
    "eventDatePlustwodaysandtwelvehours": ISODate("2023-04-05T00:00:00Z)"
  },
  {
    "_id": 3,
    "eventDate": "2023-04-03T14:00:00Z",
    "eventDatePlustwodaysandtwelvehours": ISODate("2023-04-06T02:00:00Z)"
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$dateAdd` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
```

```
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('events');

  const result = await collection.aggregate([
    {
      $project: {
        _id: 1,
        eventDate: 1,
        eventDatePlustwodaysandtwelvehours: {
          $dateAdd: {
            startDate: {
              $dateAdd: {
                startDate: "$eventDate",
                unit: "day",
                amount: 2
              }
            },
            unit: "hour",
            amount: 12
          }
        }
      }
    }
  ]).toArray();

  console.log(result);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient
from bson.date_time import datetime

def example():
```

```
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['events']

    result = list(collection.aggregate([
        {
            '$project': {
                '_id': 1,
                'eventDate': 1,
                'eventDatePlustwodaysandtwelvehours': {
                    '$dateAdd': {
                        'startDate': {
                            '$dateAdd' : {
                                'startDate': '$eventDate',
                                'unit': 'day',
                                'amount': 2,
                            }
                        },
                        'unit': 'hour',
                        'amount': 12,
                    }
                }
            }
        }
    ]))

    print(result)

    client.close()

example()
```

$DateDiff

Neu ab Version 5.0

Wird von Elastic Cluster nicht unterstützt.

Der `$dateDiff` Aggregationsoperator berechnet die Differenz zwischen zwei Daten in angegebenen Einheiten. Er gibt die Anzahl der Einheitengrenzen zurück, die zwischen dem Start- und dem Enddatum überschritten wurden.

Parameter

- `startDate`: Der Ausdruck für das Anfangsdatum.
- `endDate`: Der Ausdruck für das Enddatum.
- `unit`: Die Zeiteinheit für die Differenz. Unterstützte Einheiten sind `year`,`quarter`,`month`,`week`,`day`,`hour`,`minute`,`second`, und`millisecond`.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$dateDiff` Operator verwendet wird, um die Anzahl der Tage zwischen Auftragserteilung und Lieferung zu berechnen.

Erstellen Sie Beispieldokumente

```
db.shipments.insertMany([
  {
    orderId: 1001,
    orderDate: ISODate("2025-01-10T08:00:00Z"),
    deliveryDate: ISODate("2025-01-15T14:30:00Z")
  },
  {
    orderId: 1002,
    orderDate: ISODate("2025-02-05T10:00:00Z"),
    deliveryDate: ISODate("2025-02-12T16:45:00Z")
  }
]);
```

Beispiel für eine Abfrage

```
db.shipments.aggregate([
  {
    $project: {
      orderId: 1,
      orderDate: 1,
      deliveryDate: 1,
      daysToDeliver: {
        $dateDiff: {
```

```
          startDate: "$orderDate",
          endDate: "$deliveryDate",
          unit: "day"
        }
      }
    }
  }
]);
```

Ausgabe

```
[
  {
    _id: ObjectId('6924a5f2d66dcae121d29517'),
    orderId: 1001,
    orderDate: ISODate('2025-01-10T08:00:00.000Z'),
    deliveryDate: ISODate('2025-01-15T14:30:00.000Z'),
    daysToDeliver: 5
  },
  {
    _id: ObjectId('6924a5f2d66dcae121d29518'),
    orderId: 1002,
    orderDate: ISODate('2025-02-05T10:00:00.000Z'),
    deliveryDate: ISODate('2025-02-12T16:45:00.000Z'),
    daysToDeliver: 7
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$dateDiff` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const shipments = db.collection('shipments');
```

```
  const result = await shipments.aggregate([
    {
      $project: {
        orderId: 1,
        orderDate: 1,
        deliveryDate: 1,
        daysToDeliver: {
          $dateDiff: {
            startDate: "$orderDate",
            endDate: "$deliveryDate",
            unit: "day"
          }
        }
      }
    }
  ]).toArray();

  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    shipments = db['shipments']

    result = list(shipments.aggregate([
        {
            "$project": {
                "orderId": 1,
                "orderDate": 1,
                "deliveryDate": 1,
                "daysToDeliver": {
                    "$dateDiff": {
```

```
                        "startDate": "$orderDate",
                        "endDate": "$deliveryDate",
                        "unit": "day"
                    }
                }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$datum FromParts

Neu ab Version 5.0.1 und 8.0

Der `$dateFromParts` Aggregationsoperator in Amazon DocumentDB erstellt einen Datums- und Uhrzeitwert aus seinen Bestandteilen, wie Jahr, Monat, Tag, Stunde, Minute, Sekunde und Millisekunde, und gibt ihn zurück.

Parameter

- `year`: Das Kalenderjahr. Erforderlich, wenn keine Teile nach ISO 8601 verwendet werden.
- `month`,`day`,, `hour` `minute`,`second`,`millisecond`: (optional) Die verbleibenden Datums- und Uhrzeitkomponenten. Ausgelassene Komponenten verwenden standardmäßig ihren niedrigsten gültigen Wert.
- `isoWeekYear`,`isoWeek`,`isoDayOfWeek`: (optional) ISO 8601-Datumskomponenten. Verwenden Sie diese anstelle von`year`, und`month`, `day` um ein Datum aus einem ISO-8601-Wochendatum zu konstruieren.
- `timezone`: (optional) Die Zeitzone, die bei der Erstellung des Datums verwendet werden soll. Wenn nicht angegeben, wird UTC verwendet.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$dateFromParts` Operator verwendet wird, um ein Datum aus einzelnen Komponenten zu erstellen.

Erstellen Sie Beispieldokumente

```
db.parts.insertMany([
  { _id: 1, y: 2023, m: 4, d: 1 },
  { _id: 2, y: 2023, m: 12, d: 25 }
]);
```

Beispiel für eine Abfrage

```
db.parts.aggregate([
  {
    $project: {
      _id: 1,
      constructedDate: {
        $dateFromParts: { year: "$y", month: "$m", day: "$d" }
      }
    }
  }
])
```

Ausgabe

```
[
  { "_id": 1, "constructedDate": ISODate("2023-04-01T00:00:00Z") },
  { "_id": 2, "constructedDate": ISODate("2023-12-25T00:00:00Z") }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$dateFromParts` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
```

```
  const collection = db.collection('parts');

  const result = await collection.aggregate([
    {
      $project: {
        _id: 1,
        constructedDate: {
          $dateFromParts: { year: "$y", month: "$m", day: "$d" }
        }
      }
    }
  ]).toArray();

  console.log(result);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['parts']

    result = list(collection.aggregate([
        {
            '$project': {
                '_id': 1,
                'constructedDate': {
                    '$dateFromParts': { 'year': '$y', 'month': '$m', 'day': '$d' }
                }
            }
        }
    ]))

    print(result)
```

```
    client.close()

example()
```

$datum FromString

Der `$dateFromString` Aggregationsoperator in Amazon DocumentDB ermöglicht es Ihnen, eine Datums- und Uhrzeitzeichenfolge in ein Datumsobjekt zu konvertieren. Dies ist nützlich, wenn Ihre Daten als Zeichenketten gespeichert sind, Sie aber datumsbasierte Operationen mit den Daten durchführen müssen.

Parameter

- `dateString`: Eine Zeichenfolge, die ein Datum und eine Uhrzeit darstellt.
- `format`: (optional) Eine Zeichenfolge, die das Format von angibt`dateString`. Falls nicht angegeben, versucht Amazon DocumentDB, die Zeichenfolge im ISO-8601 Format zu analysieren.
- `timezone`: (optional) Eine Zeichenfolge, die die Zeitzone angibt. Falls nicht angegeben, verwendet Amazon DocumentDB die Zeitzone des Servers.
- `onError`: (optional) Gibt die Aktion an, die ausgeführt werden soll, wenn die Konvertierung fehlschlägt. Mögliche Werte sind `'error'` (der Standardwert, der einen Fehler auslöst), `'null'` (Rückgabe`null`) oder `'replace'` (ersetzt den Wert durch die in der `onErrorMessage` Option angegebene Ersatzzeichenfolge).
- `onErrorMessage`: (optional) Wenn auf gesetzt `onError` ist`'replace'`, gibt diese Option die Ersatzzeichenfolge an.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie Sie `$dateFromString` eine Datumszeichenfolge in Amazon DocumentDB in ein Datumsobjekt konvertieren können.

Erstellen Sie Beispieldokumente

```
db.missionLog.insertMany([
{ _id: 1, event: "missionStart", logDate: "2020-03-15T13:41:33"},
{ _id: 2, event: "jumpPoint1", logDate: "2020-03-15T13:45:34"},
{ _id: 3, event: "jumpPoint2", logDate: "2020-03-15T13:48:21"},
{ _id: 4, event: "jumpPoint3", logDate: "2020-03-15T13:52:09"},
```

```
{ _id: 5, event: "missionEnd", logDate: "2020-03-15T13:58:44"}
]);
```

Beispiel für eine Abfrage

```
db.missionLog.aggregate([
  {
    $project: {
      event: '$event',
      logDate: {
        $dateFromString: {
          dateString: '$logDate'
        }
      }
    }
  }
]);
```

Ausgabe

```
[
  {
    "_id": 1,
    "event": "missionStart",
    "logDate": ISODate("2020-03-15T13:41:33Z")
  },
  {
    "_id": 2,
    "event": "jumpPoint1",
    "logDate": ISODate("2020-03-15T13:45:34Z")
  },
  {
    "_id": 3,
    "event": "jumpPoint2",
    "logDate": ISODate("2020-03-15T13:48:21Z")
  },
  {
    "_id": 4,
    "event": "jumpPoint3",
    "logDate": ISODate("2020-03-15T13:52:09Z")
  },
  {
    "_id": 5,
```

```
    "event": "missionEnd",
    "logDate": ISODate("2020-03-15T13:58:44Z")
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$dateFromString` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('missionLog');

  const result = await collection.aggregate([
    {
      $project: {
        event: '$event',
        logDate: {
          $dateFromString: {
            dateString: '$logDate'
          }
        }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient
```

```
def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['missionLog']

    result = list(collection.aggregate([
        {
            '$project': {
                'event': '$event',
                'logDate': {
                    '$dateFromString': {
                        'dateString': '$logDate'
                    }
                }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$DateSubtrahieren

Neu ab Version 5.0

Mit dem `$dateSubtract` Aggregationsoperator in Amazon DocumentDB können Sie eine angegebene Dauer von einem Datumswert subtrahieren.

Parameter

- `date`: Ein Datumsausdruck, der in ein Datum oder einen Zeitstempel aufgelöst wird.
- `subtrahend`: Ein Ausdruck für die Dauer, der angibt, wie viel Zeit vom Ausdruck abgezogen werden soll. `date`
- `unit`: Eine Zeichenfolge, die die Zeiteinheit für den `subtrahend` Ausdruck angibt. Unterstützte Einheiten sind „Jahr“, „Quartal“, „Monat“, „Woche“, „Tag“, „Stunde“, „Minute“, „Sekunde“ und „Millisekunde“.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$dateSubtract` Operator verwendet wird, um das Datum vor einem Jahr anhand des aktuellen Datums zu berechnen.

Erstellen Sie Beispieldokumente

```
db.events.insertOne({
  eventName: "Player joined",
  eventTime: ISODate("2023-04-01T12:00:00Z")
});
```

Beispiel für eine Abfrage

```
db.events.aggregate([
  {
    $project: {
      eventName: 1,
      oneYearAgo: {
        $dateSubtract: {
          startDate: "$eventTime",
          amount: 1,
          unit: "year"
        }
      }
    }
  }
])
```

Ausgabe

```
{
  "_id" : ObjectId("64567890abcdef012345678"),
  "eventName" : "Player joined",
  "oneYearAgo" : ISODate("2022-04-01T12:00:00Z")
}
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$dateSubtract` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const events = db.collection('events');

  const result = await events.aggregate([
    {
      $project: {
        eventName: 1,
        oneYearAgo: {
          $dateSubtract: {
            startDate: "$eventTime",
            amount: 1,
            unit: "year"
          }
        }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient
from bson.date_time import datetime

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    events = db['events']
```

```
    result = list(events.aggregate([
        {
            '$project': {
                'eventName': 1,
                'oneYearAgo': {
                    '$dateSubtract': {
                        'startDate': '$eventTime',
                        'amount': 1,
                        'unit': 'year'
                    }
                }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$datum ToParts

Neu ab Version 5.0.1 und 8.0

Der `$dateToParts` Aggregationsoperator in Amazon DocumentDB gibt ein Dokument zurück, das die Bestandteile eines Datums- und Uhrzeitwerts enthält, z. B. Jahr, Monat, Tag, Stunde, Minute, Sekunde und Millisekunde.

Parameter

- `date`: Der Datums- und Uhrzeitwert, der in Teile aufgeteilt werden soll.
- `timezone`: (optional) Die Zeitzone, die beim Extrahieren der Datumsteile verwendet werden soll. Wenn nicht angegeben, werden die Teile in UTC zurückgegeben.
- `iso8601`: (optional) Wenn auf gesetzt`true`, verwendet das zurückgegebene Dokument ISO-8601-Datumsteile (`isoWeekYear`,`isoWeek`,`isoDayOfWeek`). Der Standardwert ist `false`.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$dateToParts` Operator verwendet wird, um die einzelnen Teile eines Datums zurückzugeben.

Erstellen Sie Beispieldokumente

```
db.events.insertMany([
  { _id: 1, eventDate: ISODate("2023-04-01T10:30:15Z") },
  { _id: 2, eventDate: ISODate("2023-12-25T18:45:00Z") }
]);
```

Beispiel für eine Abfrage

```
db.events.aggregate([
  {
    $project: {
      _id: 1,
      parts: { $dateToParts: { date: "$eventDate" } }
    }
  }
])
```

Ausgabe

```
[
  {
    "_id": 1,
    "parts": { "year": 2023, "month": 4, "day": 1, "hour": 10, "minute": 30, "second":
 15, "millisecond": 0 }
  },
  {
    "_id": 2,
    "parts": { "year": 2023, "month": 12, "day": 25, "hour": 18, "minute": 45,
 "second": 0, "millisecond": 0 }
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$dateToParts` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('events');

  const result = await collection.aggregate([
    {
      $project: {
        _id: 1,
        parts: { $dateToParts: { date: "$eventDate" } }
      }
    }
  ]).toArray();

  console.log(result);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['events']

    result = list(collection.aggregate([
        {
            '$project': {
                '_id': 1,
                'parts': { '$dateToParts': { 'date': '$eventDate' } }
```

```
            }
        }
    ]))

    print(result)

    client.close()

example()
```

$datum ToString

Der `$dateToString` Aggregationsoperator in Amazon DocumentDB wird verwendet, um einen Datums- oder Zeitstempelwert in eine Zeichenkettendarstellung zu konvertieren. Dies ist nützlich, wenn Sie Datum und Uhrzeit für die Anzeige oder Weiterverarbeitung auf eine bestimmte Weise formatieren müssen.

Parameter

- `date`: Der Datums- oder Zeitstempelwert, der in eine Zeichenfolge konvertiert werden soll.
- `format`: Eine Zeichenfolge, die das Format angibt, in dem das Datum dargestellt werden soll. Die Formatzeichenfolge kann verschiedene Formatbezeichner enthalten, z. B. `%Y` für das vierstellige Jahr, `%m` für den zweistelligen Monat, `%d` für den zweistelligen Tag des Monats usw.
- `timezone`: (optional) Die Zeitzone, die für die Konvertierung verwendet werden soll. Falls nicht angegeben, wird die Zeitzone des Servers verwendet, der den Amazon DocumentDB-Cluster hostet.
- `onNull`: (optional) Der Wert, der zurückgegeben werden soll, wenn der `date` Parameter `null`

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung des `$dateToString` Operators zur Formatierung des `logDate` Felds der `missionLog` Sammlung.

Erstellen Sie Beispieldokumente

```
db.missionLog.insertMany([
  { _id: 1, "event":"missionStart", logDate: new Date("2020-03-15T13:41:33Z") },
  { _id: 2, "event":"jumpPoint1", logDate: new Date("2020-03-15T13:45:34Z") },
  { _id: 3, "event":"jumpPoint2", logDate: new Date("2020-03-15T13:48:21Z") },
```

```
  { _id: 4, "event":"jumpPoint3", logDate: new Date("2020-03-15T13:52:09Z") },
  { _id: 5, "event":"missionEnd", logDate: new Date("2020-03-15T13:58:44Z") }
]);
```

Beispiel für eine Abfrage

```
db.missionLog.aggregate([
  {
    $project: {
      event: "$event",
      logDateFormatted: {
        $dateToString: {
          format: "%Y-%m-%d %H:%M:%S",
          date: "$logDate"
        }
      }
    }
  }
])
```

Ausgabe

```
[
  {
    "_id": 1,
    "event": "missionStart",
    "logDateFormatted": "2020-03-15 13:41:33"
  },
  {
    "_id": 2,
    "event": "jumpPoint1",
    "logDateFormatted": "2020-03-15 13:45:34"
  },
  {
    "_id": 3,
    "event": "jumpPoint2",
    "logDateFormatted": "2020-03-15 13:48:21"
  },
  {
    "_id": 4,
    "event": "jumpPoint3",
    "logDateFormatted": "2020-03-15 13:52:09"
  },
```

```
  {
    "_id": 5,
    "event": "missionEnd",
    "logDateFormatted": "2020-03-15 13:58:44"
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$dateToString` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

Hier ist ein Beispiel für die Verwendung des `$dateToString` Operators in einer Node.js Anwendung:

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('missionLog');

  const result = await collection.aggregate([
    {
      $project: {
        event: "$event",
        logDateFormatted: {
          $dateToString: {
            format: "%Y-%m-%d %H:%M:%S",
            date: "$logDate"
          }
        }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}
```

```
example();
```

Python

Hier ist ein Beispiel für die Verwendung des `$dateToString` Operators in einer Python-Anwendung:

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['missionLog']

    pipeline = [
        {
            '$project': {
                'event': '$event',
                'logDateFormatted': {
                    '$dateToString': {
                        'format': '%Y-%m-%d %H:%M:%S',
                        'date': '$logDate'
                    }
                }
            }
        }
    ]

    result = list(collection.aggregate(pipeline))
    print(result)
    client.close()

example()
```

$DateTrunc

Neu ab Version 8.0

Wird von Elastic Cluster nicht unterstützt.

Der `$dateTrunc` Aggregationsoperator in Amazon DocumentDB kürzt ein Datum auf eine angegebene Einheit.

Parameter

- `date`: Ein Datumsausdruck, der in ein Datum oder einen Zeitstempel aufgelöst wird.
- `unit`: Eine Zeichenfolge, die die Zeiteinheit für den Subtrahend-Ausdruck angibt. Unterstützte Einheiten sind`year`,`quarter`,`month`,`week`,, `day` `hour`, `minute`, `second` und. `millisecond`

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$dateTrunc` Operator verwendet wird, um ein Datum auf die Stunde zu kürzen.

Erstellen Sie Beispieldokumente

```
db.events.insertMany([
  {
    eventName: "Event 1",
    eventTime: ISODate("2025-04-01T12:15:00Z")
  },
  {
    eventName: "Event 2",
    eventTime: ISODate("2025-08-15T14:33:22Z")
  },
]);
```

Beispiel für eine Abfrage

```
db.events.aggregate([
  {
    $project: {
      eventName: 1,
      eventTime: 1,
      truncatedToHour: {
        $dateTrunc: {
          date: "$eventTime",
          unit: "hour"
        }
      }
    }
  }
```

```
]);
```

Ausgabe

```
[
  {
    _id: ObjectId('6924a258d66dcae121d29515'),
    eventName: 'Event 1',
    eventTime: ISODate('2025-04-01T12:15:00.000Z'),
    truncatedToHour: ISODate('2025-04-01T12:00:00.000Z')
  },
  {
    _id: ObjectId('6924a258d66dcae121d29516'),
    eventName: 'Event 2',
    eventTime: ISODate('2025-08-15T14:33:22.000Z'),
    truncatedToHour: ISODate('2025-08-15T14:00:00.000Z')
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$dateTrunc` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const events = db.collection('events');

  const result = await events.aggregate([
    {
      $project: {
        eventName: 1,
        eventTime: 1,
        truncatedToHour: {
          $dateTrunc: {
            date: "$eventTime",
```

```
            unit: "hour"
          }
        }
      }
    }
  ]).toArray();

  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    events = db['events']

    result = list(events.aggregate([
        {
            "$project": {
                "eventName": 1,
                "eventTime": 1,
                "truncatedToHour": {
                    "$dateTrunc": {
                        "date": "$eventTime",
                        "unit": "hour"
                    }
                }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$Tag OfMonth

Der `$dayOfMonth` Aggregationsoperator in Amazon DocumentDB ruft den Tag des Monats (von 1 bis 31) für ein bestimmtes Datum ab. Dieser Operator ist nützlich, um den Tag des Monats aus Datumsfeldern in Ihren Dokumenten zu gruppieren, zu filtern oder zu extrahieren.

Parameter

- `date expression`: Der Datumsausdruck kann ein Datumsfeld aus dem Dokument, ein Datumsobjekt oder eine Datumszeichenfolge sein.

Beispiel (MongoDB Shell)

Dieses Beispiel zeigt, wie der `$dayOfMonth` Operator verwendet wird, um den Tag des Monats aus einem Datumsfeld im Dokument zu extrahieren.

Erstellen Sie Beispieldokumente

```
db.events.insertMany([
  { _id: 1, eventDate: new Date("2022-01-15T12:00:00Z") },
  { _id: 2, eventDate: new Date("2022-02-28T15:30:00Z") },
  { _id: 3, eventDate: new Date("2022-03-01T09:45:00Z") },
  { _id: 4, eventDate: new Date("2022-04-30T23:59:59Z") }
]);
```

Beispiel für eine Abfrage

```
db.events.aggregate([
  { $project: {
    eventDay: { $dayOfMonth: "$eventDate" }
  }}
])
```

Ausgabe

```
[
  { "_id" : 1, "eventDay" : 15 },
  { "_id" : 2, "eventDay" : 28 },
  { "_id" : 3, "eventDay" : 1 },
  { "_id" : 4, "eventDay" : 30 }
```

```
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$dayOfMonth` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const events = db.collection('events');

  const result = await events.aggregate([
    { $project: {
      eventDay: { $dayOfMonth: "$eventDate" }
    }}
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.mydatabase
    events = db.events

    result = list(events.aggregate([
        { "$project": {
```

```
            "eventDay": { "$dayOfMonth": "$eventDate" }
        }}
    ]))

    print(result)
    client.close()

example()
```

$Tag OfWeek

Der `$dayOfWeek` Operator in Amazon DocumentDB extrahiert den Wochentag aus einem bestimmten Datumsfeld. Es gibt den Wochentag als Zahl zwischen 1 (Sonntag) und 7 (Samstag) zurück, was dem Verhalten in MongoDB entspricht.

Parameter

- `date field`: Das Datumsfeld, aus dem der Wochentag extrahiert werden soll.

Beispiel (MongoDB Shell)

Dieses Beispiel zeigt, wie der `$dayOfWeek` Operator verwendet wird, um den Wochentag aus dem `date` Feld in der `weather` Sammlung zu extrahieren.

Erstellen Sie Beispieldokumente

```
db.weather.insertMany([
  {
    "temperature": 97.5,
    "humidity": 0.60,
    "date": new Date("2023-04-01")
  },
  {
    "temperature": 95.2,
    "humidity": 0.55,
    "date": new Date("2023-04-02")
  },
  {
    "temperature": 92.8,
    "humidity": 0.65,
    "date": new Date("2023-04-03")
```

```
  }
]);
```

Beispiel für eine Abfrage

```
db.weather.aggregate([
  {
    $project: {
      dayOfWeek: { $dayOfWeek: "$date" }
    }
  }
]).pretty();
```

Ausgabe

```
{ "_id" : ObjectId("64272c6663f4f8ce422c2d91"), "dayOfWeek" : 7 }
{ "_id" : ObjectId("64272c6663f4f8ce422c2d92"), "dayOfWeek" : 1 }
{ "_id" : ObjectId("64272c6663f4f8ce422c2d93"), "dayOfWeek" : 2 }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$dayOfWeek` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('weather');

  const result = await collection.aggregate([
    {
      $project: {
        dayOfWeek: { $dayOfWeek: '$date' }
      }
    }
```

```
  ]).toArray();

  console.log(result);

  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.weather

    result = list(collection.aggregate([
        {
            '$project': {
                'dayOfWeek': { '$dayOfWeek': '$date' }
            }
        }
    ]))

    print(result)

    client.close()

if __name__ == '__main__':
    main()
```

$Tag OfYear

Der `$dayOfYear` Operator in Amazon DocumentDB gibt den Tag des Jahres für ein Datum als Zahl zwischen 1 und 366 zurück (365 in Nicht-Schaltjahren).

Parameter

- `expression`: Das Datumsfeld oder der Ausdruck, aus dem der Tag des Jahres extrahiert werden soll.

Beispiel (MongoDB Shell)

Dieses Beispiel zeigt, wie der `$dayOfYear` Operator verwendet wird, um den Tag des Jahres aus einem Datumsfeld in einer Amazon DocumentDB-Sammlung zu extrahieren.

Erstellen Sie Beispieldokumente

```
db.weather.insert([
  {
    "temperature" : 97.5,
    "humidity": 0.60,
    "date" : new Date("2023-03-15")
  },
  {
    "temperature" : 82.3,
    "humidity": 0.75,
    "date" : new Date("2023-12-31")
  }
])
```

Beispiel für eine Abfrage

```
db.weather.aggregate([
  {
    $project: {
      dayOfYear: { $dayOfYear: "$date" }
    }
  }
]).pretty()
```

Ausgabe

```
{ "_id" : ObjectId("642b86fc7d8e07af279bbe63"), "dayOfYear" : 74 }
{ "_id" : ObjectId("642b86fc7d8e07af279bbe64"), "dayOfYear" : 365 }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$dayOfYear` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('weather');

  const result = await collection.aggregate([
    {
      $project: {
        dayOfYear: { $dayOfYear: "$date" }
      }
    }
  ]).toArray();

  console.log(result);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['weather']

    result = list(collection.aggregate([
```

```
        {
            '$project': {
                'dayOfYear': { '$dayOfYear': '$date' }
            }
        }
    ]))

    print(result)

    client.close()

example()
```

$ Grad ToRadians

Neu ab Version 8.0.1.

Der `$degreesToRadians` Operator in Amazon DocumentDB konvertiert einen Wert von Grad in Bogenmaß.

Parameter

- `expression`: Ein Ausdruck, der in einen numerischen Wert in Grad aufgelöst wird.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$degreesToRadians` Operator verwendet wird, um Gradwerte in Bogenmaß umzurechnen.

Erstellen Sie Beispieldokumente

```
db.angles.insertMany([
  { "_id": 1, "degrees": 0 },
  { "_id": 2, "degrees": 90 },
  { "_id": 3, "degrees": 180 },
  { "_id": 4, "degrees": 360 }
]);
```

Beispiel für eine Abfrage

```
db.angles.aggregate([
```

```
  { $project: {
    "radians": { $degreesToRadians: "$degrees" }
  }}
]);
```

Ausgabe

```
[
  { "_id": 1, "radians": 0 },
  { "_id": 2, "radians": 1.5707963267948966 },
  { "_id": 3, "radians": 3.141592653589793 },
  { "_id": 4, "radians": 6.283185307179586 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$degreesToRadians` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('angles');

  const result = await collection.aggregate([
    { $project: {
      "radians": { $degreesToRadians: "$degrees" }
    }}
  ]).toArray();

  console.log(result);
  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['angles']

    result = list(collection.aggregate([
        { '$project': {
            'radians': { '$degreesToRadians': '$degrees' }
        }}
    ]))

    print(result)
    client.close()

if __name__ == "__main__":
    main()
```

$ Dichter Rang

Neu ab Version 8.0.2.

Der `$denseRank` Fensteroperator in Amazon DocumentDB gibt die Position jedes Dokuments innerhalb seiner Partition entsprechend der Reihenfolge der `$setWindowFields` Phasen `sortBy` zurück. Dokumente mit demselben `sortBy` Wert erhalten denselben Rang, und der nächste eindeutige Wert verwendet die unmittelbar darauf folgende Position, ohne Lücken zu hinterlassen (z. B.`1, 2, 2, 3`). Sie wird unter dem `output` Feld der `$setWindowFields` Stufe angegeben.

Es ist ein Fensteroperator, der nur auf der `$setWindowFields` Bühne verwendet wird.

Parameter

- `$denseRank`nimmt ein leeres Dokument `{}` als Wert an.
- Es akzeptiert keinen `window` Frame.
- Es wird genau ein `sortBy` Schlüssel auf der `$setWindowFields` Bühne benötigt.

Beispiel (MongoDB Shell)

Das folgende Beispiel partitioniert die Dokumente nach`game`, sortiert jede Partition `score` in absteigender Reihenfolge danach und weist jedem Spieler einen hohen Rang zu.

Erstellen Sie Beispieldokumente

```
db.players.insertMany([
  { _id: 1, game: "A", player: "Alice", score: 90 },
  { _id: 2, game: "A", player: "Bob", score: 85 },
  { _id: 3, game: "A", player: "Carol", score: 85 },
  { _id: 4, game: "A", player: "Dave", score: 70 },
  { _id: 5, game: "B", player: "Erin", score: 95 },
  { _id: 6, game: "B", player: "Frank", score: 80 }
]);
```

Beispiel für eine Abfrage

```
db.players.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$game",
      sortBy: { score: -1 },
      output: {
        denseRank: { $denseRank: {} }
      }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "game": "A", "player": "Alice", "score": 90, "denseRank": 1 },
  { "_id": 2, "game": "A", "player": "Bob", "score": 85, "denseRank": 2 },
  { "_id": 3, "game": "A", "player": "Carol", "score": 85, "denseRank": 2 },
  { "_id": 4, "game": "A", "player": "Dave", "score": 70, "denseRank": 3 },
  { "_id": 5, "game": "B", "player": "Erin", "score": 95, "denseRank": 1 },
  { "_id": 6, "game": "B", "player": "Frank", "score": 80, "denseRank": 2 }
]
```

Innerhalb jeder `game` Partition werden die Spieler absteigend sortiert`score`. Bob und Carol liegen gleichauf `85` und teilen sich den Rang`2`, und Dave folgt auf Rang `3` ohne Lücke.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$denseRank` Operators in der `$setWindowFields` Phase anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  try {
    await client.connect();
    const db = client.db('test');
    const players = db.collection('players');

    await players.insertMany([
      { _id: 1, game: "A", player: "Alice", score: 90 },
      { _id: 2, game: "A", player: "Bob", score: 85 },
      { _id: 3, game: "A", player: "Carol", score: 85 },
      { _id: 4, game: "A", player: "Dave", score: 70 },
      { _id: 5, game: "B", player: "Erin", score: 95 },
      { _id: 6, game: "B", player: "Frank", score: 80 }
    ]);

    const result = await players.aggregate([
      {
        $setWindowFields: {
          partitionBy: "$game",
          sortBy: { score: -1 },
          output: {
            denseRank: { $denseRank: {} }
          }
        }
      }
    ]).toArray();
```

```
    console.log(result);
  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    try:
        db = client['test']
        players = db['players']

        players.insert_many([
            { '_id': 1, 'game': 'A', 'player': 'Alice', 'score': 90 },
            { '_id': 2, 'game': 'A', 'player': 'Bob', 'score': 85 },
            { '_id': 3, 'game': 'A', 'player': 'Carol', 'score': 85 },
            { '_id': 4, 'game': 'A', 'player': 'Dave', 'score': 70 },
            { '_id': 5, 'game': 'B', 'player': 'Erin', 'score': 95 },
            { '_id': 6, 'game': 'B', 'player': 'Frank', 'score': 80 }
        ])

        result = list(players.aggregate([
            {
                '$setWindowFields': {
                    'partitionBy': '$game',
                    'sortBy': { 'score': -1 },
                    'output': {
                        'denseRank': { '$denseRank': {} }
                    }
                }
            }
        ]))

        print(result)
```

```
    finally:
        client.close()

example()
```

$divide

Der `$divide` Operator in der Amazon DocumentDB-Aggregationspipeline wird verwendet, um eine Zahl durch eine andere zu dividieren. Es ist ein nützlicher Operator für die Ausführung mathematischer Operationen an numerischen Feldern in Ihren Dokumenten.

Parameter

- `numerator`: Die Dividende oder die Zahl, die geteilt werden soll.
- `denominator`: Der Divisor oder die Zahl, durch die dividiert werden soll.

Beispiel (MongoDB Shell)

Dieses Beispiel zeigt, wie der `$divide` Operator verwendet wird, um den Stundensatz für Mitarbeiter auf der Grundlage ihres Jahresgehalts und der Anzahl der Arbeitsstunden pro Jahr zu berechnen.

Erstellen Sie Beispieldokumente

```
db.employees.insertMany([
  { _id: 1, name: "John Doe", salary: 80000, hoursPerYear: 2080 },
  { _id: 2, name: "Jane Smith", salary: 90000, hoursPerYear: 2080 },
  { _id: 3, name: "Bob Johnson", salary: 75000, hoursPerYear: 2080 }
]);
```

Beispiel für eine Abfrage

```
db.employees.aggregate([
  {
    $project: {
      name: 1,
      hourlyRate: { $divide: ["$salary", "$hoursPerYear"] }
    }
  }
])
```

Ausgabe

```
[
  { "_id" : 1, "name" : "John Doe", "hourlyRate" : 38.46153846153846 },
  { "_id" : 2, "name" : "Jane Smith", "hourlyRate" : 43.26923076923077 },
  { "_id" : 3, "name" : "Bob Johnson", "hourlyRate" : 36.05769230769231 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$divide` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function calculateHourlyRate() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const employees = db.collection('employees');

  const result = await employees.aggregate([
    {
      $project: {
        name: 1,
        hourlyRate: { $divide: ["$salary", "$hoursPerYear"] }
      }
    }
  ]).toArray();

  console.log(result);
  client.close();
}

calculateHourlyRate();
```

Python

```
from pymongo import MongoClient
```

```
def calculate_hourly_rate():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.mydatabase
    employees = db.employees

    result = list(employees.aggregate([
        {
            '$project': {
                'name': 1,
                'hourlyRate': { '$divide': ['$salary', '$hoursPerYear'] }
            }
        }
    ]))

    print(result)
    client.close()

calculate_hourly_rate()
```

$ Dokumentnummer

Neu ab Version 8.0.2.

Der `$documentNumber` Fensteroperator in Amazon DocumentDB gibt für jedes Dokument in seiner Partition eine eindeutige fortlaufende Nummer zurück, die der Reihenfolge der `$setWindowFields` Phasen entspricht. `sortBy` Im Gegensatz zu `$rank` und `$denseRank` behandelt er Verknüpfungen nicht speziell: Jedes Dokument erhält eine eindeutige Nummer (z. B.`1, 2, 3, 4`), und Dokumente mit gleichen `sortBy` Werten werden in der Reihenfolge nummeriert, in der sie gefunden werden. Sie wird unter dem `output` Feld der `$setWindowFields` Stufe angegeben.

Es ist ein Fensteroperator, der nur auf der `$setWindowFields` Bühne verwendet wird.

Parameter

- `$documentNumber`nimmt ein leeres Dokument `{}` als Wert an.
- Es akzeptiert keinen `window` Frame.
- Es wird genau ein `sortBy` Schlüssel auf der `$setWindowFields` Bühne benötigt.

Beispiel (MongoDB Shell)

Das folgende Beispiel partitioniert die Dokumente nach`game`, sortiert jede Partition `score` in absteigender Reihenfolge und weist jedem Player eine fortlaufende Dokumentnummer zu.

Erstellen Sie Beispieldokumente

```
db.players.insertMany([
  { _id: 1, game: "A", player: "Alice", score: 90 },
  { _id: 2, game: "A", player: "Bob", score: 85 },
  { _id: 3, game: "A", player: "Carol", score: 85 },
  { _id: 4, game: "A", player: "Dave", score: 70 },
  { _id: 5, game: "B", player: "Erin", score: 95 },
  { _id: 6, game: "B", player: "Frank", score: 80 }
]);
```

Beispiel für eine Abfrage

```
db.players.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$game",
      sortBy: { score: -1 },
      output: {
        documentNumber: { $documentNumber: {} }
      }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "game": "A", "player": "Alice", "score": 90, "documentNumber": 1 },
  { "_id": 2, "game": "A", "player": "Bob", "score": 85, "documentNumber": 2 },
  { "_id": 3, "game": "A", "player": "Carol", "score": 85, "documentNumber": 3 },
  { "_id": 4, "game": "A", "player": "Dave", "score": 70, "documentNumber": 4 },
  { "_id": 5, "game": "B", "player": "Erin", "score": 95, "documentNumber": 1 },
  { "_id": 6, "game": "B", "player": "Frank", "score": 80, "documentNumber": 2 }
]
```

Innerhalb jeder `game` Partition erhält jeder Spieler eine eindeutige fortlaufende Nummer in absteigender Reihenfolge`score`. Bob und Carol stehen gleichauf, sind 85 aber immer noch nummeriert 2 und zwar 3 in der Reihenfolge, in der sie angetroffen werden.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$documentNumber` Operators in der `$setWindowFields` Phase anzuzeigen, wählen Sie den Tab für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  try {
    await client.connect();
    const db = client.db('test');
    const players = db.collection('players');

    await players.insertMany([
      { _id: 1, game: "A", player: "Alice", score: 90 },
      { _id: 2, game: "A", player: "Bob", score: 85 },
      { _id: 3, game: "A", player: "Carol", score: 85 },
      { _id: 4, game: "A", player: "Dave", score: 70 },
      { _id: 5, game: "B", player: "Erin", score: 95 },
      { _id: 6, game: "B", player: "Frank", score: 80 }
    ]);

    const result = await players.aggregate([
      {
        $setWindowFields: {
          partitionBy: "$game",
          sortBy: { score: -1 },
          output: {
            documentNumber: { $documentNumber: {} }
          }
        }
      }
```

```
    ]).toArray();

    console.log(result);
  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    try:
        db = client['test']
        players = db['players']

        players.insert_many([
            { '_id': 1, 'game': 'A', 'player': 'Alice', 'score': 90 },
            { '_id': 2, 'game': 'A', 'player': 'Bob', 'score': 85 },
            { '_id': 3, 'game': 'A', 'player': 'Carol', 'score': 85 },
            { '_id': 4, 'game': 'A', 'player': 'Dave', 'score': 70 },
            { '_id': 5, 'game': 'B', 'player': 'Erin', 'score': 95 },
            { '_id': 6, 'game': 'B', 'player': 'Frank', 'score': 80 }
        ])

        result = list(players.aggregate([
            {
                '$setWindowFields': {
                    'partitionBy': '$game',
                    'sortBy': { 'score': -1 },
                    'output': {
                        'documentNumber': { '$documentNumber': {} }
                    }
                }
            }
        ]))
```

```
        print(result)
    finally:
        client.close()

example()
```

$eq

Der `$eq` Aggregationsoperator vergleicht zwei Werte und gibt zurück, `true` wenn sie gleich sind, andernfalls gibt er zurück`false`.

Parameter

- `expression1`: Der erste zu vergleichende Wert.
- `expression2`: Der zweite zu vergleichende Wert.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$eq` Operator verwendet wird, um zu überprüfen, ob die Produktmengen den Zielwerten entsprechen.

Erstellen Sie Beispieldokumente

```
db.inventory.insertMany([
  { _id: 1, item: "Widget", qty: 50, target: 50 },
  { _id: 2, item: "Gadget", qty: 30, target: 50 },
  { _id: 3, item: "Tool", qty: 50, target: 40 }
]);
```

Beispiel für eine Abfrage

```
db.inventory.aggregate([
  {
    $project: {
      item: 1,
      qty: 1,
      target: 1,
      meetsTarget: { $eq: ["$qty", "$target"] }
    }
```

```
  }
]);
```

Ausgabe

```
[
  { _id: 1, item: 'Widget', qty: 50, target: 50, meetsTarget: true },
  { _id: 2, item: 'Gadget', qty: 30, target: 50, meetsTarget: false },
  { _id: 3, item: 'Tool', qty: 50, target: 40, meetsTarget: false }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$eq` Aggregationsoperators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('inventory');

  const result = await collection.aggregate([
    {
      $project: {
        item: 1,
        qty: 1,
        target: 1,
        meetsTarget: { $eq: ["$qty", "$target"] }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}
```

```
example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['inventory']

    result = list(collection.aggregate([
        {
            '$project': {
                'item': 1,
                'qty': 1,
                'target': 1,
                'meetsTarget': { '$eq': ['$qty', '$target'] }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$exp

Neu ab Version 4.0

Mit dem `$exp` Operator in Amazon DocumentDB können Sie die Konstante e auf eine bestimmte Zahl erhöhen.

Parameter

- `expression`: Der auszuwertende Ausdruck. Dies kann ein beliebiger gültiger Aggregationsausdruck sein, einschließlich Feldverweise, arithmetischer Operationen und anderer Aggregationsstufen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$exp` Operator verwendet wird, um alle Dokumente zu finden, bei denen das `quantity` Feld größer als das `price` Feld ist.

Erstellen Sie Beispieldokumente

```
db.items.insertMany([
  { item: "canvas", quantity: 4 },
  { item: "journal", quantity: 2 }
]);
```

Beispiel für eine Abfrage

```
db.items.aggregate([
  { $project: {
    "quantityRaised": {$exp: "$quantity"}}
  }
]);
```

Ausgabe

```
[
  {
    _id: ObjectId('6920b785311cf98b79d2950d'),
    quantityRaised: 54.598150033144236
  },
  {
    _id: ObjectId('6920b785311cf98b79d2950e'),
    quantityRaised: 7.38905609893065
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$exp` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');
```

```
async function aggregateExp() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const items = db.collection('items');

  const result = await items.aggregate([
    { $project: {
      "quantityRaised": {$exp: "$quantity"}}
    }
    ]).toArray();

  console.log(result);
  client.close();
}

aggregateExp();
```

Python

```
from pymongo import MongoClient

def aggregate_exp():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    items = db.items

    result = list(items.aggregate([
      { "$project": {
        "quantityRaised": {"$exp": "$quantity"}}
      }
    ]))

    print(result)
    client.close()

aggregate_exp()
```

$facet

Neu ab Version 8.0.2.

Die `$facet` Aggregationsphase in Amazon DocumentDB verarbeitet mehrere Aggregationspipelines innerhalb einer einzigen Stufe für denselben Satz von Eingabedokumenten. Jedes Ausgabefeld definiert seine eigene Pipeline, und jede Pipeline erhält dieselben Eingabedokumente. Auf diese Weise können Sie mehrere unabhängige Aggregationen, wie Anzahlen, Gruppierungen und zusammenfassende Statistiken, in einer einzigen Abfrage berechnen, anstatt die Sammlung durch mehrere separate Abfragen ausführen zu müssen.

Jede Subpipeline läuft unabhängig und erzeugt eine Reihe von Dokumenten, die in dem entsprechenden Ausgabefeld gespeichert werden. Da die Sub-Pipelines keinen gemeinsamen Status haben, wirkt sich die Ausgabe einer Pipeline nicht auf eine andere aus.

Syntax

```
{
  $facet: {
    <outputField1>: [ <stage1>, <stage2>, ... ],
    <outputField2>: [ <stage1>, <stage2>, ... ],
    ...
  }
}
```

Parameter

- `outputField`(erforderlich): Der Name eines Ausgabefeldes, das das Ergebnisarray der zugehörigen Aggregationspipeline enthält. Sie können ein oder mehrere Ausgabefelder angeben. Jedes Ausgabefeld ist einer Reihe von Aggregationspipeline-Stufen zugeordnet, die auf die Eingabedokumente angewendet werden.

Ausgabe

Die `$facet` Stufe gibt ein einzelnes Dokument zurück. Jedes von Ihnen angegebene Ausgabefeld enthält eine Reihe von Dokumenten, die von der entsprechenden Subpipeline erstellt wurden.

Einschränkungen

- Eine `$facet` Subpipeline darf die folgenden Stufen nicht enthalten:`$changeStream`,`$collStats`, `$facet` (Verschachtelung ist nicht

zulässig)$geoNear,$indexStats,$listSearchIndexes,,$merge, $out $planCacheStats$search, $searchMeta und. $vectorSearch

- Das Dokument, das von der $facet Phase zurückgegeben wird, muss innerhalb der Größenbeschränkung von 16 MB für BSON-Dokumente bleiben.
- Eine $sort Stufe, die vor einer $facet Stufe liegt, kann nicht die Reihenfolge eines Akkumulators ($first,, $last $push$firstN, oder$mergeObjects) in einer$lastN, $bucketAuto O-Stufe bestimmen $group$bucket, wenn diese Reihenfolge über die Grenze hinweg angewendet werden müsste. $facet Dies gilt unabhängig davon, ob sich die Gruppierungsphase innerhalb einer $facet Unterleitung oder hinter der Stufe befindet. $facet In diesem Fall gibt Amazon DocumentDB einen Fehler zurück, anstatt die Pipeline auszuführen. Um einen solchen Akkumulator zu bestellen, platzieren Sie die $sort Stufe in der $facet Subpipeline, unmittelbar vor der Gruppierungsphase. Order-insensitive Akkumulatoren wie,$sum, und $min maxcount, sind nicht betroffen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der $facet Stagingbereich verwendet wird, um mehrere Aggregationen für dieselben Verkaufsdaten in einer einzigen Abfrage zu berechnen.

Erstellen Sie Beispieldokumente

```
db.sales.insertMany([
  { item: "abc", price: 10, quantity: 2, date: new Date("2020-09-01") },
  { item: "def", price: 20, quantity: 1, date: new Date("2020-10-01") },
  { item: "ghi", price: 5, quantity: 3, date: new Date("2020-11-01") },
  { item: "jkl", price: 15, quantity: 2, date: new Date("2020-12-01") },
  { item: "abc", price: 25, quantity: 1, date: new Date("2021-01-01") }
]);
```

Beispiel für eine Abfrage

Der folgende Befehl wird verwendet, $facet um in einem einzigen Schritt die gesamte verkaufte Menge pro Artikel, die Gesamtpreisstatistiken und die Gesamtzahl der Dokumente zu berechnen:

```
db.sales.aggregate([
  {
    $facet: {
      "totalByItem": [
        { $group: { _id: "$item", totalQuantity: { $sum: "$quantity" } } }
```

```
      ],
      "priceStats": [
        { $group: { _id: null, avgPrice: { $avg: "$price" }, maxPrice: { $max:
 "$price" } } }
      ],
      "documentCount": [
        { $count: "total" }
      ]
    }
  }
])
```

Ausgabe

Der Befehl gibt die folgende Ausgabe zurück:

```
[
  {
    totalByItem: [
      { _id: "abc", totalQuantity: 3 },
      { _id: "def", totalQuantity: 1 },
      { _id: "ghi", totalQuantity: 3 },
      { _id: "jkl", totalQuantity: 2 }
    ],
    priceStats: [
      { _id: null, avgPrice: 15, maxPrice: 25 }
    ],
    documentCount: [
      { total: 5 }
    ]
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$facet` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
```

```
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const sales = db.collection('sales');

  const result = await sales.aggregate([
    {
      $facet: {
        "totalByItem": [
          { $group: { _id: "$item", totalQuantity: { $sum: "$quantity" } } }
        ],
        "priceStats": [
          { $group: { _id: null, avgPrice: { $avg: "$price" }, maxPrice: { $max:
 "$price" } } }
        ],
        "documentCount": [
          { $count: "total" }
        ]
      }
    }
  ]).toArray();

  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    sales = db['sales']

    result = list(sales.aggregate([
        {
            '$facet': {
```

```
                'totalByItem': [
                    {'$group': {'_id': '$item', 'totalQuantity': {'$sum':
 '$quantity'}}}
                ],
                'priceStats': [
                    {'$group': {'_id': None, 'avgPrice': {'$avg': '$price'},
 'maxPrice': {'$max': '$price'}}}
                ],
                'documentCount': [
                    {'$count': 'total'}
                ]
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$filter

Der `$filter` Operator in Amazon DocumentDB wird verwendet, um einen Filterausdruck auf jedes Element eines Arrays anzuwenden und ein Array zurückzugeben, das nur die Elemente enthält, die der angegebenen Bedingung entsprechen. Dieser Operator ist nützlich, wenn Sie komplexe Filteroperationen für Array-Felder in Ihren Dokumenten durchführen müssen.

Parameter

- `input`: Das zu filternde Array-Feld.
- `as`: Der Variablenname, der für jedes Element des `input` Arrays innerhalb des `cond` Ausdrucks verwendet werden soll.
- `cond`: Der boolesche Ausdruck, der bestimmt, ob ein bestimmtes Element in das Ausgabe-Array aufgenommen werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$filter` Operator verwendet wird, um den Kunden jeder Bestellung zu projizieren und ein neues Array-Feld PaidItems zu erstellen, das nur die Artikel aus

dem Artikel-Array enthält, deren Preis größer als 15 ist. Im Wesentlichen werden die Artikel jeder Bestellung so gefiltert, dass nur Produkte aufgenommen werden, die mehr als 15€ kosten.

Erstellen Sie Beispieldokumente

```
db.orders.insertMany([
  { _id: 1, customer: "abc123", items: [
    { name: "Product A", price: 10, qty: 2 },
    { name: "Product B", price: 20, qty: 1 }
  ]},
  { _id: 2, customer: "def456", items: [
    { name: "Product C", price: 5, qty: 3 },
    { name: "Product D", price: 15, qty: 4 }
  ]},
  { _id: 3, customer: "ghi789", items: [
    { name: "Product E", price: 8, qty: 3 },
    { name: "Product F", price: 12, qty: 1 }
  ]}
]);
```

Beispiel für eine Abfrage

```
db.orders.aggregate([
  {
    $project: {
      customer: 1,
      paidItems: {
        $filter: {
          input: "$items",
          as: "item",
          cond: { $gt: ["$$item.price", 15] }
        }
      }
    }
  }
]).pretty();
```

Ausgabe

```
[
  {
    _id: 1,
    customer: 'abc123',
```

```
    paidItems: [ { name: 'Product B', price: 20, qty: 1 } ]
  },
  { _id: 2, customer: 'def456', paidItems: [] },
  { _id: 3, customer: 'ghi789', paidItems: [] }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$filter` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const uri = 'mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false';
  const client = new MongoClient(uri);

  try {
    await client.connect();

    const db = client.db('test');
    const collection = db.collection('orders');

    const result = await collection.aggregate([
      {
        $project: {
          customer: 1,
          paidItems: {
            $filter: {
              input: "$items",
              as: "item",
              cond: { $gt: ["$$item.price", 15] }
            }
          }
        }
      }
    ]).toArray();

    console.log(JSON.stringify(result, null, 2));
```

```
  } catch (error) {
    console.error('Error:', error);
  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient
from pprint import pprint

def example():
    uri = 'mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false'

    with MongoClient(uri) as client:
        db = client.test
        collection = db.orders

        result = list(collection.aggregate([
            {
                '$project': {
                    'customer': 1,
                    'paidItems': {
                        '$filter': {
                            'input': '$items',
                            'as': 'item',
                            'cond': { '$gt': ['$$item.price', 15] }
                        }
                    }
                }
            }
        ]))

        for doc in result:
            pprint(doc)

example()
```

$first

Neu ab Version 5.0.

Wird von Elastic Cluster nicht unterstützt.

Der `$first` Operator in Amazon DocumentDB gibt das erste Dokument aus einer gruppierten Gruppe von Dokumenten zurück. Es wird häufig in Aggregationspipelines verwendet, um das erste Dokument abzurufen, das einer bestimmten Bedingung entspricht.

Parameter

- `expression`: Der Ausdruck, der als erster Wert in jeder Gruppe zurückgegeben werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel veranschaulicht die Verwendung des `$first` Operators zum Abrufen des ersten Elementwerts, der bei der Aggregation für jede Kategorie gefunden wurde.

Hinweis: `$first` Gibt das erste Dokument auf der Grundlage der aktuellen Reihenfolge der Dokumente in der Pipeline zurück. Um eine bestimmte Reihenfolge sicherzustellen (z. B. nach Datum, Preis usw.), sollte der `$sort` `$group` Phase eine Stufe vorangestellt werden.

Erstellen Sie Musterdokumente

```
db.products.insertMany([
  { _id: 1, item: "abc", price: 10, category: "food" },
  { _id: 2, item: "jkl", price: 20, category: "food" },
  { _id: 3, item: "xyz", price: 5, category: "toy" },
  { _id: 4, item: "abc", price: 5, category: "toy" }
]);
```

Beispiel für eine Abfrage

```
db.products.aggregate([
  { $group: { _id: "$category", firstItem: { $first: "$item" } } }
]);
```

Ausgabe

```
[
  { "_id" : "food", "firstItem" : "abc" },
```

```
  { "_id" : "toy", "firstItem" : "xyz" }
]
```

Beispiel für die Verwendung von Fensteroperatoren (MongoDB Shell)

Neu ab Version 8.0.2.

Der `$first` Operator kann auch als Fensterbediener auf der `$setWindowFields` Bühne verwendet werden. In diesem Zusammenhang gibt es den Wert von `expression` aus dem ersten Dokument in jedem Fenster zurück. Sie geben den Operator unter dem `output` Feld an und definieren optional die Fenstergrenzen mit einem `window` Dokument.

> **Note**
>
> Bei Verwendung als Fensteroperator in `$setWindowFields` `$first` ist er auf 100 MB an Zwischendaten begrenzt. Eine Operation, die dieses Limit überschreitet, gibt einen Fehler zurück.

Erstellen Sie Beispieldokumente

```
db.stockPrices.insertMany([
  { _id: 1, ticker: "ABC", hour: 1, price: 50 },
  { _id: 2, ticker: "ABC", hour: 2, price: 45 },
  { _id: 3, ticker: "ABC", hour: 3, price: 60 },
  { _id: 4, ticker: "ABC", hour: 4, price: 40 },
  { _id: 5, ticker: "XYZ", hour: 1, price: 30 },
  { _id: 6, ticker: "XYZ", hour: 2, price: 35 },
  { _id: 7, ticker: "XYZ", hour: 3, price: 25 }
]);
```

Beispiel für eine Abfrage

Das folgende Beispiel partitioniert die Dokumente nach`ticker`, sortiert jede Partition nach und gibt den ersten (frühesten) Preis zurück`hour`, der in jeder Partition aufgezeichnet wurde.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
```

```
      output: {
        firstPrice: {
          $first: "$price",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "ticker": "ABC", "hour": 1, "price": 50, "firstPrice": 50 },
  { "_id": 2, "ticker": "ABC", "hour": 2, "price": 45, "firstPrice": 50 },
  { "_id": 3, "ticker": "ABC", "hour": 3, "price": 60, "firstPrice": 50 },
  { "_id": 4, "ticker": "ABC", "hour": 4, "price": 40, "firstPrice": 50 },
  { "_id": 5, "ticker": "XYZ", "hour": 1, "price": 30, "firstPrice": 30 },
  { "_id": 6, "ticker": "XYZ", "hour": 2, "price": 35, "firstPrice": 30 },
  { "_id": 7, "ticker": "XYZ", "hour": 3, "price": 25, "firstPrice": 30 }
]
```

Jedes Dokument wird um den Preis des ersten Dokuments in seinem Fenster erweitert (dem frühesten `hour` in der Partition). `firstPrice`

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$first` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten. Die folgenden Beispiele zeigen sowohl die Verwendung des Akkumulators (in`$group`) als auch die Verwendung von Fensteroperatoren (in`$setWindowFields`):

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const uri = 'mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false';
  const client = new MongoClient(uri);
```

```
try {
  await client.connect();

  const db = client.db('test');

  // Accumulator usage: first value per group
  const products = db.collection('products');
  await products.insertMany([
    { _id: 1, item: "abc", price: 10, category: "food" },
    { _id: 2, item: "jkl", price: 20, category: "food" },
    { _id: 3, item: "xyz", price: 5, category: "toy" },
    { _id: 4, item: "abc", price: 5, category: "toy" }
  ]);
  const accumulatorResult = await products.aggregate([
    { $group: { _id: "$category", firstItem: { $first: "$item" } } }
  ]).toArray();
  console.log('Accumulator result:', accumulatorResult);

  // Window operator usage: first value within each partition
  const stockPrices = db.collection('stockPrices');
  await stockPrices.insertMany([
    { _id: 1, ticker: "ABC", hour: 1, price: 50 },
    { _id: 2, ticker: "ABC", hour: 2, price: 45 },
    { _id: 3, ticker: "ABC", hour: 3, price: 60 },
    { _id: 4, ticker: "ABC", hour: 4, price: 40 },
    { _id: 5, ticker: "XYZ", hour: 1, price: 30 },
    { _id: 6, ticker: "XYZ", hour: 2, price: 35 },
    { _id: 7, ticker: "XYZ", hour: 3, price: 25 }
  ]);
  const windowResult = await stockPrices.aggregate([
    {
      $setWindowFields: {
        partitionBy: "$ticker",
        sortBy: { hour: 1 },
        output: {
          firstPrice: {
            $first: "$price",
            window: { documents: ["unbounded", "current"] }
          }
        }
      }
    }
  ]).toArray();
  console.log('Window result:', windowResult);
```

```
  } catch (error) {
    console.error('Error:', error);
  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient
from pprint import pprint

def example():
    client = None
    try:
        client = MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

        db = client['test']

        # Accumulator usage: first value per group
        products = db['products']
        products.insert_many([
            { '_id': 1, 'item': 'abc', 'price': 10, 'category': 'food' },
            { '_id': 2, 'item': 'jkl', 'price': 20, 'category': 'food' },
            { '_id': 3, 'item': 'xyz', 'price': 5, 'category': 'toy' },
            { '_id': 4, 'item': 'abc', 'price': 5, 'category': 'toy' }
        ])
        accumulator_result = list(products.aggregate([
            { '$group': { '_id': '$category', 'firstItem': { '$first': '$item' } } }
        ]))
        pprint(accumulator_result)

        # Window operator usage: first value within each partition
        stock_prices = db['stockPrices']
        stock_prices.insert_many([
            { '_id': 1, 'ticker': 'ABC', 'hour': 1, 'price': 50 },
            { '_id': 2, 'ticker': 'ABC', 'hour': 2, 'price': 45 },
            { '_id': 3, 'ticker': 'ABC', 'hour': 3, 'price': 60 },
```

```
            { '_id': 4, 'ticker': 'ABC', 'hour': 4, 'price': 40 },
            { '_id': 5, 'ticker': 'XYZ', 'hour': 1, 'price': 30 },
            { '_id': 6, 'ticker': 'XYZ', 'hour': 2, 'price': 35 },
            { '_id': 7, 'ticker': 'XYZ', 'hour': 3, 'price': 25 }
        ])
        window_result = list(stock_prices.aggregate([
            {
                '$setWindowFields': {
                    'partitionBy': '$ticker',
                    'sortBy': { 'hour': 1 },
                    'output': {
                        'firstPrice': {
                            '$first': '$price',
                            'window': { 'documents': ['unbounded', 'current'] }
                        }
                    }
                }
            }
        ]))
        pprint(window_result)

    except Exception as e:
        print(f"An error occurred: {e}")

    finally:
        if client:
            client.close()

example()
```

$FirstN

Neu ab Version 8.0.1.

Der `$firstN` Operator in Amazon DocumentDB gibt die ersten N Elemente zurück. Wenn er als Akkumulator in einer `$group` Stufe verwendet wird, gibt er ein Array der ersten N Werte in jeder Gruppe zurück. Wenn er als Array-Ausdrucksoperator verwendet wird, gibt er die ersten N Elemente eines Arrays zurück.

Parameter

- `input`: Der Ausdruck, der in das Feld oder Array aufgelöst wird, aus dem Werte zurückgegeben werden sollen.
- n: Eine positive Ganzzahl, die angibt, wie viele Werte zurückgegeben werden sollen. Bei Verwendung als Akkumulator in einer `$group` Phase n kann es sich auch um einen Ausdruck handeln, sofern er auf der Grundlage des `_id` Gruppenfeldes in eine positive Ganzzahl aufgelöst wird.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$firstN` Akkumulator verwendet wird, um die ersten beiden Mengen für jedes Element während der Aggregation abzurufen.

> **Note**
>
> `$firstN`wählt Werte in der Reihenfolge aus, in der Dokumente die `$group` Bühne erreichen. Um die ersten N Werte für eine bestimmte Reihenfolge zurückzugeben (z. B. nach Datum oder Punktzahl), fügen Sie eine `$sort` Stufe davor hinzu`$group`.

Erstellen Sie Beispieldokumente

```
db.sales.insertMany([
  { item: "abc", quantity: 10, date: ISODate("2023-01-01") },
  { item: "abc", quantity: 5, date: ISODate("2023-01-02") },
  { item: "abc", quantity: 8, date: ISODate("2023-01-03") },
  { item: "xyz", quantity: 15, date: ISODate("2023-01-01") },
  { item: "xyz", quantity: 7, date: ISODate("2023-01-02") },
  { item: "xyz", quantity: 3, date: ISODate("2023-01-03") }
]);
```

Beispiel für eine Abfrage

```
db.sales.aggregate([
  { $group: { _id: "$item", firstTwoQuantities: { $firstN: { input: "$quantity", n:
 2 } } } }
]);
```

Ausgabe

```
[
  { "_id": "abc", "firstTwoQuantities": [10, 5] },
  { "_id": "xyz", "firstTwoQuantities": [15, 7] }
]
```

Beispiel für die Verwendung von Ausdrücken (MongoDB Shell)

Der `$firstN` Operator kann auch als Ausdruck innerhalb einer `$project` Stufe verwendet werden, um die ersten N Elemente eines Array-Feldes zurückzugeben.

Erstellen Sie Beispieldokumente

```
db.inventory.insertMany([
  { _id: 1, item: "abc", tags: ["red", "green", "blue", "yellow", "purple"] },
  { _id: 2, item: "xyz", tags: ["alpha", "beta", "gamma"] }
]);
```

Beispiel für eine Abfrage

```
db.inventory.aggregate([
  { $project: {
      firstThreeTags: { $firstN: { input: "$tags", n: 3 } }
    }}
]);
```

Ausgabe

```
[
  { "_id": 1, "firstThreeTags": ["red", "green", "blue"] },
  { "_id": 2, "firstThreeTags": ["alpha", "beta", "gamma"] }
]
```

Beispiel für die Verwendung von Fensteroperatoren (MongoDB Shell)

Neu ab Version 8.0.2.

Der `$firstN` Operator kann auch als Fensterbediener auf der `$setWindowFields` Bühne verwendet werden. In diesem Zusammenhang gibt er ein Array mit bis zu den ersten n Werten für die Dokumente in jedem Fenster zurück. Sie geben den Operator unter dem `output` Feld an und definieren optional die Fenstergrenzen mit einem `window` Dokument.

> **Note**
>
> Bei Verwendung als Fensteroperator in `$setWindowFields` `$firstN` ist er auf 100 MB an Zwischendaten begrenzt. Eine Operation, die dieses Limit überschreitet, gibt einen Fehler zurück.

Erstellen Sie Beispieldokumente

```
db.stockPrices.insertMany([
  { _id: 1, ticker: "ABC", hour: 1, price: 50 },
  { _id: 2, ticker: "ABC", hour: 2, price: 45 },
  { _id: 3, ticker: "ABC", hour: 3, price: 60 },
  { _id: 4, ticker: "ABC", hour: 4, price: 40 },
  { _id: 5, ticker: "XYZ", hour: 1, price: 30 },
  { _id: 6, ticker: "XYZ", hour: 2, price: 35 },
  { _id: 7, ticker: "XYZ", hour: 3, price: 25 }
]);
```

Beispiel für eine Abfrage

Das folgende Beispiel partitioniert die Dokumente nach`ticker`, sortiert jede Partition nach und gibt die ersten beiden Preise zurück`hour`, die vom Anfang der Partition bis zum aktuellen Dokument angezeigt werden.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        firstTwoSoFar: {
          $firstN: { input: "$price", n: 2 },
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "ticker": "ABC", "hour": 1, "price": 50, "firstTwoSoFar": [50] },
  { "_id": 2, "ticker": "ABC", "hour": 2, "price": 45, "firstTwoSoFar": [50, 45] },
  { "_id": 3, "ticker": "ABC", "hour": 3, "price": 60, "firstTwoSoFar": [50, 45] },
  { "_id": 4, "ticker": "ABC", "hour": 4, "price": 40, "firstTwoSoFar": [50, 45] },
  { "_id": 5, "ticker": "XYZ", "hour": 1, "price": 30, "firstTwoSoFar": [30] },
  { "_id": 6, "ticker": "XYZ", "hour": 2, "price": 35, "firstTwoSoFar": [30, 35] },
  { "_id": 7, "ticker": "XYZ", "hour": 3, "price": 25, "firstTwoSoFar": [30, 35] }
]
```

Jedes Dokument wird um die ersten beiden `price` Werte innerhalb seines Fensters erweitert (in der `hour` Reihenfolge). `firstTwoSoFar`

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$firstN` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten. Die folgenden Beispiele zeigen die Verwendung von Akkumulatoren (in`$group`), Ausdrücken (in`$project`) und Fensteroperatoren (in`$setWindowFields`):

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  try {
    await client.connect();
    const db = client.db('test');

    // Accumulator usage: first N values per group
    const sales = db.collection('sales');
    await sales.insertMany([
      { item: "abc", quantity: 10, date: new Date("2023-01-01") },
      { item: "abc", quantity: 5, date: new Date("2023-01-02") },
      { item: "abc", quantity: 8, date: new Date("2023-01-03") },
      { item: "xyz", quantity: 15, date: new Date("2023-01-01") },
      { item: "xyz", quantity: 7, date: new Date("2023-01-02") },
      { item: "xyz", quantity: 3, date: new Date("2023-01-03") }
```

```
    ]);
    const accumulatorResult = await sales.aggregate([
      { $group: { _id: "$item", firstTwoQuantities: { $firstN: { input: "$quantity",
n: 2 } } } }
    ]).toArray();
    console.log('Accumulator result:', accumulatorResult);

    // Expression usage: first N elements of an array field
    const inventory = db.collection('inventory');
    await inventory.insertMany([
      { _id: 1, item: "abc", tags: ["red", "green", "blue", "yellow", "purple"] },
      { _id: 2, item: "xyz", tags: ["alpha", "beta", "gamma"] }
    ]);
    const expressionResult = await inventory.aggregate([
      { $project: { firstThreeTags: { $firstN: { input: "$tags", n: 3 } } } }
    ]).toArray();
    console.log('Expression result:', expressionResult);

    // Window operator usage: first N values over a window in each partition
    const stockPrices = db.collection('stockPrices');
    await stockPrices.insertMany([
      { _id: 1, ticker: "ABC", hour: 1, price: 50 },
      { _id: 2, ticker: "ABC", hour: 2, price: 45 },
      { _id: 3, ticker: "ABC", hour: 3, price: 60 },
      { _id: 4, ticker: "ABC", hour: 4, price: 40 },
      { _id: 5, ticker: "XYZ", hour: 1, price: 30 },
      { _id: 6, ticker: "XYZ", hour: 2, price: 35 },
      { _id: 7, ticker: "XYZ", hour: 3, price: 25 }
    ]);
    const windowResult = await stockPrices.aggregate([
      {
        $setWindowFields: {
          partitionBy: "$ticker",
          sortBy: { hour: 1 },
          output: {
            firstTwoSoFar: {
              $firstN: { input: "$price", n: 2 },
              window: { documents: ["unbounded", "current"] }
            }
          }
        }
      }
    ]).toArray();
    console.log('Window result:', windowResult);
```

```
  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient
from datetime import datetime

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    try:
        db = client['test']

        # Accumulator usage: first N values per group
        sales = db['sales']
        sales.insert_many([
            { 'item': 'abc', 'quantity': 10, 'date': datetime(2023, 1, 1) },
            { 'item': 'abc', 'quantity': 5, 'date': datetime(2023, 1, 2) },
            { 'item': 'abc', 'quantity': 8, 'date': datetime(2023, 1, 3) },
            { 'item': 'xyz', 'quantity': 15, 'date': datetime(2023, 1, 1) },
            { 'item': 'xyz', 'quantity': 7, 'date': datetime(2023, 1, 2) },
            { 'item': 'xyz', 'quantity': 3, 'date': datetime(2023, 1, 3) }
        ])
        accumulator_result = list(sales.aggregate([
            { '$group': { '_id': '$item', 'firstTwoQuantities': { '$firstN':
 { 'input': '$quantity', 'n': 2 } } } }
        ]))
        print('Accumulator result:', accumulator_result)

        # Expression usage: first N elements of an array field
        inventory = db['inventory']
        inventory.insert_many([
            { '_id': 1, 'item': 'abc', 'tags': ['red', 'green', 'blue', 'yellow',
 'purple'] },
            { '_id': 2, 'item': 'xyz', 'tags': ['alpha', 'beta', 'gamma'] }
```

```
        ])
        expression_result = list(inventory.aggregate([
            { '$project': { 'firstThreeTags': { '$firstN': { 'input': '$tags', 'n':
 3 } } } }
        ]))
        print('Expression result:', expression_result)

        # Window operator usage: first N values over a window in each partition
        stock_prices = db['stockPrices']
        stock_prices.insert_many([
            { '_id': 1, 'ticker': 'ABC', 'hour': 1, 'price': 50 },
            { '_id': 2, 'ticker': 'ABC', 'hour': 2, 'price': 45 },
            { '_id': 3, 'ticker': 'ABC', 'hour': 3, 'price': 60 },
            { '_id': 4, 'ticker': 'ABC', 'hour': 4, 'price': 40 },
            { '_id': 5, 'ticker': 'XYZ', 'hour': 1, 'price': 30 },
            { '_id': 6, 'ticker': 'XYZ', 'hour': 2, 'price': 35 },
            { '_id': 7, 'ticker': 'XYZ', 'hour': 3, 'price': 25 }
        ])
        window_result = list(stock_prices.aggregate([
            {
                '$setWindowFields': {
                    'partitionBy': '$ticker',
                    'sortBy': { 'hour': 1 },
                    'output': {
                        'firstTwoSoFar': {
                            '$firstN': { 'input': '$price', 'n': 2 },
                            'window': { 'documents': ['unbounded', 'current'] }
                        }
                    }
                }
            }
        ]))
        print('Window result:', window_result)

    finally:
        client.close()

example()
```

$floor

Neu ab Version 4.0.

Der `$floor` Operator in Amazon DocumentDB gibt die größte Ganzzahl zurück, die kleiner oder gleich der angegebenen Zahl ist. Dieser Operator ist nützlich, um numerische Werte abzurunden.

Parameter

- `expression`: Der numerische Ausdruck, der abgerundet werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$floor` Operator verwendet wird, um einen Dezimalwert auf die nächste Ganzzahl abzurunden.

Erstellen Sie Beispieldokumente

```
db.numbers.insertOne({ value: 3.14 });
```

Beispiel für eine Abfrage

```
db.numbers.aggregate([
  { $project: { _id: 0, floored: { $floor: "$value" } } }
]);
```

Ausgabe

```
{ "floored" : 3 }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$floor` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const uri = 'mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false';
  const client = new MongoClient(uri);
```

```
  try {
    await client.connect();

    const db = client.db('test');
    const collection = db.collection('numbers');

    const result = await collection.aggregate([
      { $project: { _id: 0, floored: { $floor: "$value" } } }
    ]).toArray();

    console.log(result);

  } catch (error) {
    console.error('Error:', error);
  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient
from pprint import pprint

def example():
    client = None
    try:
        client = MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

        db = client.test
        collection = db.numbers

        result = list(collection.aggregate([
            { '$project': { '_id': 0, 'floored': { '$floor': '$value' }}}
        ]))

        pprint(result)
```

```
    except Exception as e:
        print(f"An error occurred: {e}")

    finally:
        if client:
            client.close()

example()
```

$geoNear

In der `$geoNear` Aggregationsphase werden Dokumente in der Reihenfolge ihrer Nähe zu einem bestimmten Punkt zurückgegeben. Sie berechnet die Entfernung vom Punkt und bezieht die Entfernung in die Ausgabedokumente ein.

Parameter

- `near`: Der Punkt, von dem aus Entfernungen berechnet werden sollen, angegeben als GeoJSON- oder Legacy-Koordinaten.
- `distanceField`: Der Feldname zum Speichern der berechneten Entfernung.
- `spherical`: Boolescher Wert, der angibt, ob sphärische Geometrie verwendet werden soll (erforderlich für GeoJSON-Punkte).
- `maxDistance`: Optional. Maximaler Abstand vom Mittelpunkt.
- `minDistance`: Optional. Minimaler Abstand vom Mittelpunkt.
- `query`: Optional. Zusätzliche anzuwendende Filterkriterien.
- `limit`: Optional. Maximale Anzahl zurückzugebender Dokumente.
- `key`: Optional. Feld, das für Geodatenabfragen verwendet werden soll, wenn mehrere Geoindizes vorhanden sind.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie die `$geoNear` Bühne verwendet wird, um Geschäfte zu finden, die einem bestimmten Standort am nächsten sind.

Erstellen Sie Beispieldokumente

```
db.stores.createIndex({ location: "2dsphere" });
```

```
db.stores.insertMany([
  { _id: 1, name: "Store A", location: { type: "Point", coordinates: [-122.4,
 37.8] } },
  { _id: 2, name: "Store B", location: { type: "Point", coordinates: [-122.5,
 37.7] } },
  { _id: 3, name: "Store C", location: { type: "Point", coordinates: [-122.3, 37.9] } }
]);
```

Beispiel für eine Abfrage

```
db.stores.aggregate([
  {
    $geoNear: {
      near: { type: "Point", coordinates: [-122.4, 37.8] },
      distanceField: "distance",
      spherical: true
    }
  }
]);
```

Ausgabe

```
[
  { _id: 1, name: 'Store A', location: { type: 'Point', coordinates: [ -122.4,
 37.8 ] }, distance: 0 },
  { _id: 3, name: 'Store C', location: { type: 'Point', coordinates: [ -122.3,
 37.9 ] }, distance: 13877.82 },
  { _id: 2, name: 'Store B', location: { type: 'Point', coordinates: [ -122.5,
 37.7 ] }, distance: 15557.89 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung der `$geoNear` Aggregationsphase anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
```

```
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('stores');

  const result = await collection.aggregate([
    {
      $geoNear: {
        near: { type: "Point", coordinates: [-122.4, 37.8] },
        distanceField: "distance",
        spherical: true
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['stores']

    result = list(collection.aggregate([
        {
            '$geoNear': {
                'near': { 'type': 'Point', 'coordinates': [-122.4, 37.8] },
                'distanceField': 'distance',
                'spherical': True
            }
        }
    ]))
```

```
    print(result)
    client.close()

example()
```

$graphLookup

Neu ab Version 8.0.2.

Die `$graphLookup` Aggregationsphase in Amazon DocumentDB führt eine rekursive Suche in einer Sammlung durch. `$graphLookup`Beginnt bei jedem Eingabedokument mit einem oder mehreren Werten und ordnet wiederholt ein Verbindungsfeld einem Zielfeld zu. Dabei folgt man der Kette übereinstimmender Dokumente, bis keine Treffer mehr vorliegen oder eine optionale Rekursionstiefe erreicht ist. Dies ist nützlich, wenn Sie grafische oder hierarchische Daten wie organisatorische Berichtsstrukturen, Kategorienbäume oder Netzwerkbeziehungen durchsuchen möchten.

Parameter

- `from`: Die Zielsammlung, nach der rekursiv gesucht werden soll. Die `from` Sammlung muss sich in derselben Datenbank befinden.
- `startWith`: Der Ausdruck, der den oder die Werte angibt, mit denen die rekursive Suche beginnen soll. Dieser Wert wird mit dem `connectToField` der Dokumente in der `from` Sammlung verglichen. Wenn `startWith` die Auswertung zu einem Array führt, wird die Suche für alle Array-Elemente gleichzeitig ausgeführt.
- `connectFromField`: Der Feldname, dessen Wert verwendet wird, um bei der rekursiven Suche eine Verbindung zum nächsten Dokument herzustellen. Handelt es sich bei dem Wert um ein Array, wird jedes Element einzeln beim Durchlauf verfolgt.
- `connectToField`: Der Feldname in der `from` Sammlung, der mit dem Wert von abgeglichen wird. `connectFromField`
- `as`: Der Name des Array-Felds, das jedem Ausgabedokument hinzugefügt wurde. Dieses Feld enthält die Dokumente, die bei der rekursiven Suche gefunden wurden. Informationen zur Reihenfolge dieser Dokumente finden Sie unter dem Element Reihenfolge der Ergebnisse im folgenden Abschnitt Verhalten.
- `maxDepth`: Eine nicht negative Ganzzahl, die die maximale Anzahl rekursiver Iterationen angibt, die nach der ersten Suche ausgeführt werden sollen. Dieser Parameter ist optional. Mit dem Wert von wird nur `0` die erste Übereinstimmung und keine Rekursion durchgeführt. Wenn Sie diesen

Wert weglassen`maxDepth`, wird die Suche rekursiv fortgesetzt, bis keine passenden Dokumente mehr gefunden werden.

- `depthField`: Der Name des Felds, das zu jedem Dokument in dem `as` Array hinzugefügt werden soll, das die Rekursionstiefe enthält, in der das Dokument gefunden wurde. Dieser Parameter ist optional. Die Tiefe wird als a dargestellt `NumberLong` und beginnt bei Null, sodass die anfängliche Suche einer Tiefe von `0` entspricht.
- `restrictSearchWithMatch`: Ein Dokument, das eine zusätzliche Übereinstimmungsbedingung festlegt, die auf die Dokumente der Suche angewendet wird. Dieser Parameter ist optional. Die Bedingung verwendet dieselbe Abfragesyntax wie eine `$match` Stufe und kann keine Aggregationsausdrücke enthalten.

> **Note**
>
> Der `$graphLookup` Stagingbereich unterliegt einer Speicherbeschränkung von 100 MB und wird nicht unterstützt`allowDiskUse:true`. Ein Vorgang, der dieses Limit überschreitet, gibt einen Fehler zurück. Wenn die Aggregation andere Stufen umfasst, bleibt die `allowDiskUse:true` Option für diese Stufen gültig.

Behavior

- Cross-collection Traversal: Wählen Sie die Sammlung aus`from`, die Sie durchsuchen möchten. Dies kann dieselbe Sammlung sein, mit der die Pipeline läuft, oder eine andere Sammlung in derselben Datenbank.
- Reihenfolge der Ergebnisse: Die Dokumente im `as` Array werden nicht in einer garantierten Reihenfolge zurückgegeben.

Beispiel (MongoDB Shell)

Im folgenden Beispiel wird `$graphLookup` eine Berichtshierarchie für Mitarbeiter durchquert und für jeden Mitarbeiter die gesamte Kette der Manager über ihm zurückgegeben.

Die folgenden Befehle, die in der Mongosh-Shell ausgeführt werden, fügen Beispieldokumente für Mitarbeiter ein, die eine Berichtshierarchie festlegen:

```
db.employees.insertMany([
```

```
  { _id: 1, name: "Dev", reportsTo: "Eliot" },
  { _id: 2, name: "Eliot", reportsTo: "Ron" },
  { _id: 3, name: "Ron", reportsTo: "Andrew" },
  { _id: 4, name: "Andrew", reportsTo: "Asya" },
  { _id: 5, name: "Asya" }
]);
```

Die folgende Aggregation gibt die vollständige Verwaltungskette für jeden Mitarbeiter in einem neuen `reportingHierarchy` Array zurück:

```
db.employees.aggregate([
  {
    $graphLookup: {
      from: "employees",
      startWith: "$reportsTo",
      connectFromField: "reportsTo",
      connectToField: "name",
      as: "reportingHierarchy",
      depthField: "level"
    }
  }
]);
```

Dieser Vorgang gibt die folgende Ausgabe zurück:

```
[
  {
    _id: 1,
    name: 'Dev',
    reportsTo: 'Eliot',
    reportingHierarchy: [
      { _id: 2, name: 'Eliot', reportsTo: 'Ron', level: 0 },
      { _id: 3, name: 'Ron', reportsTo: 'Andrew', level: 1 },
      { _id: 4, name: 'Andrew', reportsTo: 'Asya', level: 2 },
      { _id: 5, name: 'Asya', level: 3 }
    ]
  },
  {
    _id: 2,
    name: 'Eliot',
    reportsTo: 'Ron',
    reportingHierarchy: [
      { _id: 3, name: 'Ron', reportsTo: 'Andrew', level: 0 },
```

```
      { _id: 4, name: 'Andrew', reportsTo: 'Asya', level: 1 },
      { _id: 5, name: 'Asya', level: 2 }
    ]
  },
  {
    _id: 3,
    name: 'Ron',
    reportsTo: 'Andrew',
    reportingHierarchy: [
      { _id: 4, name: 'Andrew', reportsTo: 'Asya', level: 0 },
      { _id: 5, name: 'Asya', level: 1 }
    ]
  },
  {
    _id: 4,
    name: 'Andrew',
    reportsTo: 'Asya',
    reportingHierarchy: [
      { _id: 5, name: 'Asya', level: 0 }
    ]
  },
  {
    _id: 5,
    name: 'Asya',
    reportingHierarchy: []
  }
]
```

Jedes Mitarbeiterdokument erhält ein `reportingHierarchy` Array, das die rekursiv abgeglichenen Managerdokumente enthält. Das `level` Feld zeichnet die Rekursionstiefe auf, bei der jeder Manager gefunden wurde. Asya hat keinen `reportsTo` Wert, daher ist ihr `reportingHierarchy` Array leer.

Beispiel mit MaxDepth und Restrict SearchWithMatch (MongoDB Shell)

Das folgende Beispiel durchquert ein Netzwerk von Verbindungen zwischen Personen und verwendet, um die Rekursion `maxDepth` zu begrenzen und nur Dokumente `restrictSearchWithMatch` einzubeziehen, die eine zusätzliche Bedingung erfüllen. Bei der Suche werden die Verbindungen einer Startperson mit einer Tiefe von bis zu zwei Ebenen gefunden, wobei nur Personen angezeigt werden, deren `hobbies` Array Folgendes enthält. `golf`

Die folgenden Befehle, die in der Mongosh-Shell ausgeführt werden, fügen Beispieldokumente ein, in denen jede Person `friends` durch eine eindirektionale Verbindungskette nach vorne zeigt:

```
db.people.insertMany([
  { _id: 1, name: "Tanya", friends: [ "Carole", "Shirley" ], hobbies: [ "tennis",
 "golf" ] },
  { _id: 2, name: "Carole", friends: [ "Joseph" ], hobbies: [ "archery", "golf" ] },
  { _id: 3, name: "Joseph", friends: [ "Angelo" ], hobbies: [ "tennis", "golf" ] },
  { _id: 4, name: "Angelo", friends: [ "Miguel" ], hobbies: [ "travel", "golf" ] },
  { _id: 5, name: "Miguel", friends: [], hobbies: [ "golf" ] },
  { _id: 6, name: "Shirley", friends: [], hobbies: [ "frisbee" ] }
]);
```

Die folgende Aggregation beginnt bei Tanya, durchquert ihr bis zu zwei Ebenen tiefes Netzwerk von Verbindungen und führt nur die Personen, die diese Liste als Hobby auflisten: `golf`

```
db.people.aggregate([
  { $match: { name: "Tanya" } },
  {
    $graphLookup: {
      from: "people",
      startWith: "$friends",
      connectFromField: "friends",
      connectToField: "name",
      as: "golfConnections",
      maxDepth: 2,
      restrictSearchWithMatch: { hobbies: "golf" }
    }
  },
  { $project: { name: 1, "golfConnections.name": 1 } }
]);
```

Dieser Vorgang gibt die folgende Ausgabe zurück:

```
[
  {
    _id: 1,
    name: 'Tanya',
    golfConnections: [
      { name: 'Carole' },
      { name: 'Joseph' },
      { name: 'Angelo' }
    ]
  }
]
```

Nur Verbindungen, deren `hobbies` Array Folgendes enthält, `golf` sind enthalten, und das Durchlaufen wird nach zwei Rekursionsebenen beendet. Shirley ist ausgeschlossen, weil sie nichts auflistet`golf`, und Miguel ist ausgeschlossen, weil er drei Stufen von Tanya entfernt ist, hinter dem von. `maxDepth` 2 Da die Verbindungen eine unidirektionale Kette bilden, die niemals auf Tanya zurückweist, wird das Ausgangsdokument bei der rekursiven Suche nicht gefunden.

Codebeispiele

Um ein Codebeispiel anzuzeigen, das die Berichtshierarchie für Mitarbeiter anhand des ersten Beispiels mithilfe eines MongoDB-Treibers aufbaut, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  try {
    await client.connect();
    const db = client.db('test');
    const employees = db.collection('employees');

    const result = await employees.aggregate([
      {
        $graphLookup: {
          from: 'employees',
          startWith: '$reportsTo',
          connectFromField: 'reportsTo',
          connectToField: 'name',
          as: 'reportingHierarchy',
          depthField: 'level'
        }
      }
    ]).toArray();

    console.log(JSON.stringify(result, null, 2));
  } catch (err) {
    console.error('Failed to run $graphLookup aggregation:', err.message);
  } finally {
```

```
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient
from pymongo.errors import PyMongoError

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    try:
        db = client['test']
        employees = db['employees']

        pipeline = [
            {
                "$graphLookup": {
                    "from": "employees",
                    "startWith": "$reportsTo",
                    "connectFromField": "reportsTo",
                    "connectToField": "name",
                    "as": "reportingHierarchy",
                    "depthField": "level"
                }
            }
        ]

        result = list(employees.aggregate(pipeline))

        for doc in result:
            print(doc)
    except PyMongoError as error:
        print(f"Failed to run $graphLookup aggregation: {error}")
    finally:
        client.close()

example()
```

$group

Die `$group` Aggregationsphase in Amazon DocumentDB ermöglicht es Ihnen, Dokumente nach einem bestimmten Ausdruck zu gruppieren und verschiedene kumulative Operationen mit den gruppierten Daten durchzuführen. Dies kann für Aufgaben wie die Berechnung von Gesamtwerten, Durchschnittswerten oder anderen Statistiken auf der Grundlage der gruppierten Daten nützlich sein.

Parameter

- `_id`: Gibt den Ausdruck an, nach dem die Eingabedokumente gruppiert werden sollen. Dies kann ein Feldname, ein berechneter Ausdruck oder eine Kombination aus beidem sein.
- `accumulator expressions`: (optional) Ein oder mehrere Akkumulatorausdrücke, die auf die gruppierten Daten angewendet werden sollen. Diese Ausdrücke verwenden die oben genannten Akkumulatoroperatoren.

Beispiel (MongoDB Shell)

Im folgenden Beispiel werden Kunden nach ihrer Stadt gruppiert und der Gesamtbestellbetrag für jede Stadt berechnet.

Erstellen Sie Beispieldokumente

```
db.customers.insertMany([
  { name: "John Doe", city: "New York", orders: [{ amount: 100 }, { amount: 200 }] },
  { name: "Jane Smith", city: "Los Angeles", orders: [{ amount: 150 }, { amount:
 300 }] },
  { name: "Bob Johnson", city: "New York", orders: [{ amount: 75 }, { amount: 125 }] },
  { name: "Samantha Lee", city: "Chicago", orders: [{ amount: 50 }, { amount: 100 }] }
]);
```

Beispiel für eine Abfrage

```
db.customers.aggregate([
  {
    $group: {
      _id: "$city",
      totalOrders: { $sum: { $sum: "$orders.amount" } }
    }
  }
]);
```

Ausgabe

```
[
  { _id: 'Chicago', totalOrders: 150 },
  { _id: 'Los Angeles', totalOrders: 450 },
  { _id: 'New York', totalOrders: 500 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$group` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function groupByCity() {
  const uri = 'mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false';
  const client = new MongoClient(uri);

  try {
    await client.connect();
    const db = client.db('test');
    const result = await db.collection('customers').aggregate([
      { $unwind: '$orders' },
      {
        $group: {
          _id: '$city',
          totalOrders: { $sum: '$orders.amount' }
        }
      }
    ]).toArray();

    console.log(result);
  } catch (err) {
    console.error('Error during aggregation:', err);
  } finally {
    await client.close();
  }
}
```

```
groupByCity();
```

Python

```
from pymongo import MongoClient

def group_by_city():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    try:
        db = client.test
        result = list(db.customers.aggregate([
            {'$unwind': '$orders'},
            {
                '$group': {
                    '_id': '$city',
                    'totalOrders': {'$sum': '$orders.amount'}
                }
            }
        ]))
        print(result)
    except Exception as e:
        print(f"Error during aggregation: {e}")
    finally:
        client.close()

group_by_city()
```

$gt

Der `$gt` Aggregationsoperator vergleicht zwei Werte und gibt zurück, `true` wenn der erste Wert größer als der zweite ist, andernfalls kehrt `false` er zurück.

Parameter

- `expression1`: Der erste zu vergleichende Wert.
- `expression2`: Der zweite zu vergleichende Wert.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$gt` Operator verwendet wird, um Produkte zu identifizieren, die einen Preisschwellenwert überschreiten.

Erstellen Sie Beispieldokumente

```
db.products.insertMany([
  { _id: 1, name: "Laptop", price: 1200 },
  { _id: 2, name: "Mouse", price: 25 },
  { _id: 3, name: "Monitor", price: 400 }
]);
```

Beispiel für eine Abfrage

```
db.products.aggregate([
  {
    $project: {
      name: 1,
      price: 1,
      expensive: { $gt: ["$price", 100] }
    }
  }
]);
```

Ausgabe

```
[
  { _id: 1, name: 'Laptop', price: 1200, expensive: true },
  { _id: 2, name: 'Mouse', price: 25, expensive: false },
  { _id: 3, name: 'Monitor', price: 400, expensive: true }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$gt` Aggregationsoperators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');
```

```
async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('products');

  const result = await collection.aggregate([
    {
      $project: {
        name: 1,
        price: 1,
        expensive: { $gt: ["$price", 100] }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['products']

    result = list(collection.aggregate([
        {
            '$project': {
                'name': 1,
                'price': 1,
                'expensive': { '$gt': ['$price', 100] }
            }
        }
```

```
    ]))

    print(result)
    client.close()

example()
```

$gte

Der `$gte` Aggregationsoperator vergleicht zwei Werte und gibt zurück, `true` ob der erste Wert größer oder gleich dem zweiten ist, andernfalls kehrt `false` er zurück.

Parameter

- `expression1`: Der erste zu vergleichende Wert.
- `expression2`: Der zweite zu vergleichende Wert.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$gte` Operator verwendet wird, um zu überprüfen, ob Schüler eine Prüfung bestanden haben.

Erstellen Sie Beispieldokumente

```
db.students.insertMany([
  { _id: 1, name: "Alice", score: 85 },
  { _id: 2, name: "Bob", score: 60 },
  { _id: 3, name: "Charlie", score: 72 }
]);
```

Beispiel für eine Abfrage

```
db.students.aggregate([
  {
    $project: {
      name: 1,
      score: 1,
      passed: { $gte: ["$score", 70] }
    }
```

```
  }
]);
```

Ausgabe

```
[
  { _id: 1, name: 'Alice', score: 85, passed: true },
  { _id: 2, name: 'Bob', score: 60, passed: false },
  { _id: 3, name: 'Charlie', score: 72, passed: true }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$gte` Aggregationsoperators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('students');

  const result = await collection.aggregate([
    {
      $project: {
        name: 1,
        score: 1,
        passed: { $gte: ["$score", 70] }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['students']

    result = list(collection.aggregate([
        {
            '$project': {
                'name': 1,
                'score': 1,
                'passed': { '$gte': ['$score', 70] }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$hour

Der `$hour` Operator extrahiert die Stundenkomponente aus einem Datums- oder Zeitstempelfeld.

Parameter

- `dateExpression`: Das Datum, auf das der Operator angewendet wird. Dies muss ein gültiges BSON-Datum ergeben (z. B. ein Feld wie $createdAt oder ein Datumsliteral).

Ein Parameter kann auch als Dokument im folgenden Format angegeben werden:

{Datum:<dateExpression>, Zeitzone:<timezoneExpression>}

Dies ermöglicht die Anwendung zeitzonenorientierter Datumsoperationen.

```
- `<tzExpression>`: (optional) The timezone of the operation result. It must be a valid
 expression that resolves to a string formatted as either an Olson Timezone Identifier
 or a UTC Offset. If no timezone is provided, the result is in UTC.
```

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der $hour Operator verwendet wird, um die Stundenkomponente aus einem Datumsfeld zu extrahieren und die Daten entsprechend zu gruppieren.

Erstellen Sie Beispieldokumente

```
db.events.insertMany([
  { timestamp: new Date("2023-04-01T10:30:00Z") },
  { timestamp: new Date("2023-04-01T12:45:00Z") },
  { timestamp: new Date("2023-04-02T08:15:00Z") },
  { timestamp: new Date("2023-04-02T16:20:00Z") },
  { timestamp: new Date("2023-04-03T23:59:00Z") }
]);
```

Beispiel für eine Abfrage

```
db.events.aggregate([
  {
    $project: {
      hour: { $hour: "$timestamp" }
    }
  },
  {
    $group: {
      _id: "$hour",
      count: { $sum: 1 }
    }
  },
  {
    $sort: { _id: 1 }
  }
]);
```

Ausgabe

```
[
  { "_id": 8, "count": 1 },
```

```
  { "_id": 10, "count": 1 },
  { "_id": 12, "count": 1 },
  { "_id": 16, "count": 1 },
  { "_id": 23, "count": 1 }
]
```

Diese Abfrage gruppiert die Ereignisse nach der Stundenkomponente des `timestamp` Felds und zählt die Anzahl der Ereignisse für jede Stunde.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$hour` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  try {
    await client.connect();
    const db = client.db('test');
    const collection = db.collection('events');

    const result = await collection.aggregate([
      {
        $project: {
          hour: { $hour: "$timestamp" }
        }
      },
      {
        $group: {
          _id: "$hour",
          count: { $sum: 1 }
        }
      },
      {
        $sort: { _id: 1 }
      }
```

```
    ]).toArray();

    console.log(result);
  } catch (error) {
    console.error('Error occurred:', error);
  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient
from datetime import datetime

def example():
    try:
        client = MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

        db = client.test
        collection = db.events

        result = list(collection.aggregate([
            {
                "$project": {
                    "hour": {"$hour": "$timestamp"}
                }
            },
            {
                "$group": {
                    "_id": "$hour",
                    "count": {"$sum": 1}
                }
            },
            {
                "$sort": {"_id": 1}
            }
        ]))
```

```
        print(result)

    except Exception as e:
        print(f"An error occurred: {e}")

    finally:
        client.close()

example()
```

$ifNull

Der `$ifNull` Operator wird verwendet, um einen bestimmten Wert zurückzugeben, wenn der Eingabeausdruck Null oder undefiniert ergibt. Dieser Operator kann in Szenarien nützlich sein, in denen Sie einen Standardwert angeben oder Fälle behandeln null/undefined möchten.

Parameter

- `expression`: Der auszuwertende Ausdruck.
- `replacement`: Der Wert, der zurückgegeben werden soll, wenn die <`expression`> Auswertung Null oder undefiniert ergibt.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung des `$ifNull` Operators zur Bereitstellung eines Standardwerts, wenn das `name` Feld Null oder undefiniert ist.

Erstellen Sie Beispieldokumente

```
db.users.insertMany([
  { _id: 1, name: "John" },
  { _id: 2, name: null },
  { _id: 3 }
]);
```

Beispiel für eine Abfrage

```
db.users.aggregate([
  {
```

```
    $project: {
      _id: 1,
      name: { $ifNull: ["$name", "No Name"] }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "name": "John" },
  { "_id": 2, "name": "No Name" },
  { "_id": 3, "name": "No Name" }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$ifNull` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  const db = client.db('test');

  const collection = db.collection('users');

  const pipeline = [
    {
      $project: {
        _id: 1,
        name: { $ifNull: ["$name", "No Name"] }
      }
    }
  ];
```

```
  const cursor = await collection.aggregate(pipeline);

  await cursor.forEach(doc => {
    console.log(doc);
  });

  await client.close();

}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    db = client.test
    collection = db.users

    pipeline = [
        {
            "$project": {
                "_id": 1,
                "name": { "$ifNull": ["$name", "No Name"] }
            }
        }
    ]

    result = collection.aggregate(pipeline)

    for doc in result:
        print(doc)

    client.close()

example()
```

$in

Der `$in` Aggregationsoperator prüft, ob ein bestimmter Wert innerhalb eines Arrays existiert. Er gibt zurück`true`, ob der Wert im Array gefunden wurde, und `false` andernfalls.

Parameter

- `value`: Der Wert, nach dem gesucht werden soll.
- `array`: Das Array, in dem gesucht werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$in` Operator verwendet wird, um zu überprüfen, ob in den Fähigkeiten jedes Mitarbeiters eine bestimmte Fähigkeit vorhanden ist.

Erstellen Sie Beispieldokumente

```
db.employees.insertMany([
  { _id: 1, name: "Sarah", skills: ["Python", "JavaScript", "SQL"] },
  { _id: 2, name: "Mike", skills: ["Java", "C++", "Go"] },
  { _id: 3, name: "Emma", skills: ["Python", "Ruby", "Rust"] }
]);
```

Beispiel für eine Abfrage

```
db.employees.aggregate([
  {
    $project: {
      name: 1,
      hasPython: { $in: ["Python", "$skills"] }
    }
  }
]);
```

Ausgabe

```
[
  { _id: 1, name: 'Sarah', hasPython: true },
  { _id: 2, name: 'Mike', hasPython: false },
  { _id: 3, name: 'Emma', hasPython: true }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$in` Aggregationsoperators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('employees');

  const result = await collection.aggregate([
    {
      $project: {
        name: 1,
        hasPython: { $in: ["Python", "$skills"] }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['employees']

    result = list(collection.aggregate([
```

```
        {
            '$project': {
                'name': 1,
                'hasPython': { '$in': ['Python', '$skills'] }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$ index OfArray

Der `$indexOfArray` Operator in Amazon DocumentDB wird verwendet, um den Index des ersten Vorkommens eines bestimmten Elements in einem Array zu finden. Dieser Operator gibt die auf Null basierende Indexposition des ersten Elements im Array zurück, das dem angegebenen Wert entspricht. Wenn der Wert nicht gefunden wird, gibt er -1 zurück.

Parameter

- `array`: Das zu durchsuchende Array.
- `value`: Der Wert, nach dem im Array gesucht werden soll.
- `start`: (optional) Die Position im Array, von der aus die Suche gestartet werden soll. Der Standardwert lautet 0.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der OfArray $index-Operator verwendet wird, um den Index des ersten Vorkommens des Elements „Mango“ im Array „fruits“ für jedes Dokument zu finden.

Erstellen Sie Beispieldokumente

```
db.collection.insertMany([
  { _id: 1, fruits: ["apple", "banana", "cherry", "durian"] },
  { _id: 2, fruits: ["mango", "orange", "pineapple"] },
  { _id: 3, fruits: ["kiwi", "lemon", "mango"] }
]);
```

Beispiel für eine Abfrage

```
db.collection.aggregate([
  {
    $project: {
      _id: 1,
      fruitIndex: { $indexOfArray: ["$fruits", "mango"] }
    }
  }
]);
```

Ausgabe

```
{ "_id" : 1, "fruitIndex" : 1 }
{ "_id" : 2, "fruitIndex" : 0 }
{ "_id" : 3, "fruitIndex" : 2 }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$indexOfArray` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('collection');

  const result = await collection.aggregate([
    {
      $project: {
        _id: 1,
        fruitIndex: { $indexOfArray: ["$fruits", "mango"] }
      }
    }
  ]).toArray();
```

```
  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['collection']

    result = list(collection.aggregate([
        {
            '$project': {
                '_id': 1,
                'fruitIndex': { '$indexOfArray': ["$fruits", "mango"] }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$ index OfBytes

Der OfBytes $index-Operator in Amazon DocumentDB wird verwendet, um den Startindex einer Teilzeichenfolge innerhalb einer Zeichenfolge auf der Grundlage der Bytepositionen der Zeichen zu finden. Dies kann nützlich sein, wenn Sie mit Textdaten arbeiten, die Mehrbyte-Zeichen enthalten können, wie sie beispielsweise in nicht-lateinischen Schriften vorkommen.

Parameter

- `string`: Die Eingabezeichenfolge, nach der gesucht werden soll.

- `substring`: Die Teilzeichenfolge, nach der innerhalb der Eingabezeichenfolge gesucht werden soll.
- `[<start>]`: (optional) Die Startposition (auf Null basierend) der Suche. Wenn nicht angegeben, beginnt die Suche am Anfang der Zeichenfolge.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung von`$indexOfBytes`, um den Index des ersten Bindestrichs in einer Reihe von Zeichenketten zu finden, die Schreibtischpositionen darstellen.

Erstellen Sie Beispieldokumente

```
db.people.insertMany([
  { "_id": 1, "Desk": "Düsseldorf-BVV-021" },
  { "_id": 2, "Desk": "Munich-HGG-32a" },
  { "_id": 3, "Desk": "Cologne-ayu-892.50" },
  { "_id": 4, "Desk": "Dortmund-Hop-78" }
]);
```

Beispiel für eine Abfrage

```
db.people.aggregate([
  { $project: { stateLocation: { $indexOfBytes: ["$Desk", "-"] } } }
]);
```

Ausgabe

```
{ "_id" : 1, "stateLocation" : 11 }
{ "_id" : 2, "stateLocation" : 6 }
{ "_id" : 3, "stateLocation" : 7 }
{ "_id" : 4, "stateLocation" : 8 }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$indexOfBytes` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');
```

```
async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const events = db.collection('people');

  const result = await db.collection('people').aggregate([
    { $project: { stateLocation: { $indexOfBytes: ["$Desk", "-"] } } }
  ]).toArray();
  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['people']

    result = list(db.people.aggregate([
        { '$project': { 'stateLocation': { '$indexOfBytes': ['$Desk', '-'] } } }
    ]))
    print(result)
    client.close()

example()
```

$indexOfCP

Der `$indexOfCP` Operator in Amazon DocumentDB wird verwendet, um den Index in Codepunkten (CP) des ersten Vorkommens einer angegebenen Teilzeichenfolge innerhalb eines Zeichenkettenausdrucks zu finden. Dies kann beim Parsen und Extrahieren von Inhalten aus Zeichenkettenfeldern nützlich sein.

Parameter

- `string expression`: Die zu durchsuchende Zeichenfolge.
- `substring`: Die Teilzeichenfolge, nach der gesucht werden soll.
- `[<start>]`: (optional) Die Position, an der die Suche gestartet werden soll (nullbasierter Index). Standard = 0.

Beispiel (MongoDB Shell)

In diesem Beispiel verwenden wir den `$indexOfCP` Operator, um den Index des ersten Vorkommens des Bindestrichs zu ermitteln — im Feld Schreibtisch jedes Dokuments.

Erstellen Sie Beispieldokumente

```
db.people.insertMany([
  { "_id":1, "name":"John Doe", "Manager":"Jane Doe", "Role":"Developer", "Desk":
 "Düsseldorf-BVV-021"},
  { "_id":2, "name":"John Stiles", "Manager":"Jane Doe", "Role":"Manager", "Desk":
 "Munich-HGG-32a"},
  { "_id":3, "name":"Richard Roe", "Manager":"Jorge Souza", "Role":"Product", "Desk":
 "Cologne-ayu-892.50"},
  { "_id":4, "name":"Mary Major", "Manager":"Jane Doe", "Role":"Solution Architect",
 "Desk": "Dortmund-Hop-78"}
])
```

Beispiel für eine Abfrage

```
db.people.aggregate([
  { $project: { stateLocation: { $indexOfCP: [ "$Desk", "-"] } } }
])
```

Ausgabe

```
{ "_id" : 1, "stateLocation" : 10 }
{ "_id" : 2, "stateLocation" : 6 }
{ "_id" : 3, "stateLocation" : 7 }
{ "_id" : 4, "stateLocation" : 8 }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$indexOfCP` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('people');

  const result = await collection.aggregate([
    { $project: { stateLocation: { $indexOfCP: [ "$Desk", "-"] } } }
  ]).toArray();

  console.log(result);

  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
db = client['test']
collection = db['people']

result = list(collection.aggregate([
    { '$project': { 'stateLocation': { '$indexOfCP': [ '$Desk', '-' ] } } }
]))

print(result)
```

```
client.close()
```

$indexStats

Die `$indexStats` Aggregationsphase in Amazon DocumentDB bietet Einblick in die Verwendung von Indizes innerhalb einer Sammlung. Dieser Operator ermöglicht es Ihnen, die Zugriffsmuster Ihrer Indizes zu überwachen, was Ihnen helfen kann, fundierte Entscheidungen über die Indexverwaltung und -optimierung zu treffen.

Parameter

Keine

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$indexStats` Operator verwendet wird, um die Indexnutzung in einer Amazon DocumentDB-Sammlung zu analysieren.

Erstellen Sie Beispieldokumente

```
db.grocery.insertMany([
  { _id: 1, product: "milk", quantity: 10 },
  { _id: 2, product: "eggs", quantity: 20 },
  { _id: 3, product: "bread", quantity: 5 },
  { _id: 4, product: "cheese", quantity: 15 },
  { _id: 5, product: "apple", quantity: 8 }
]);
```

Beispiel für eine Abfrage

```
db.grocery.aggregate([
  { $indexStats: {} }
]);
```

Ausgabe

```
[
  {
    "name": "_id_",
    "key": {
```

```
      "_id": 1
    },
    "host": "docdb-cluster-1.cluster-123456789.us-west-2.docdb.amazonaws.com",
    "accesses": {
      "ops": NumberLong(5),
      "since": ISODate("2023-04-06T12:34:56.789Z")
    }
  },
  {
    "name": "product_1",
    "key": {
      "product": 1
    },
    "host": "docdb-cluster-1.cluster-123456789.us-west-2.docdb.amazonaws.com",
    "accesses": {
      "ops": NumberLong(10),
      "since": ISODate("2023-04-06T12:34:56.789Z")
    }
  }
]
```

In diesem Beispiel zeigt der `$indexStats` Operator, dass seit dem _id_ letzten Reset oder Serverneustart fünfmal auf den `product_1` Index zugegriffen wurde und dass zehnmal auf den Index zugegriffen wurde.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$indexStats` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function indexStats() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const result = await db.collection('grocery').aggregate([
    { $indexStats: {} }
  ]).toArray();
  console.log(result);
```

```
  await client.close();
}

indexStats();
```

Python

```
from pymongo import MongoClient

def index_stats():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    result = list(db.grocery.aggregate([
        { '$indexStats': {} }
    ]))
    print(result)
    client.close()

index_stats()
```

$isArray

Der `$isArray` Operator in Amazon DocumentDB wird verwendet, um zu prüfen, ob es sich bei einem Feld in einem Dokument um ein Array handelt. Dieser Operator kann in Aggregationspipelines und bedingten Ausdrücken nützlich sein, um Felder vom Typ Array zu verarbeiten.

Parameter

- `field`: Der Feldpfad, um zu überprüfen, ob es sich um ein Array handelt.

Beispiel (MongoDB Shell)

Dieses Beispiel zeigt, wie der `$isArray` Operator verwendet wird, um Dokumente zu identifizieren, bei denen das Feld „Inventar“ ein Array ist.

Erstellen Sie Beispieldokumente

```
db.videos.insertMany([
```

```
  { "_id":1, "name":"Live Soft", "inventory": {"Des Moines": 1000, "Ames" : 500}},
  { "_id":2, "name":"Top Pilot", "inventory": {"Mason City": 250, "Des Moines": 1000}},
  { "_id":3, "name":"Romancing the Rock", "inventory": {"Mason City": 250, "Ames" :
 500}},
  { "_id":4, "name":"Bravemind", "inventory": [{"location": "Mason City", "count":
 250}, {"location": "Des Moines", "count": 1000}, {"location": "Ames", "count": 500}]}
]);
```

Beispiel für eine Abfrage

```
db.videos.aggregate([
  {
    $match: {
      $isArray: "$inventory"
    }
  },
  {
    $project: {
      _id: 1,
      name: 1,
      "inventory.location": 1,
      "inventory.count": 1
    }
  }
]).pretty();
```

Ausgabe

```
{
  "_id": 4,
  "name": "Bravemind",
  "inventory": [
    {
      "location": "Mason City",
      "count": 250
    },
    {
      "location": "Des Moines",
      "count": 1000
    },
    {
      "location": "Ames",
      "count": 500
```

```
    }
  ]
}
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$isArray` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function run() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('videos');

  const result = await collection.aggregate([
    {
      $match: {
        $isArray: '$inventory'
      }
    },
    {
      $project: {
        _id: 1,
        name: 1,
        "inventory.location": 1,
        "inventory.count": 1
      }
    }
  ]).toArray();

  console.log(result);

  await client.close();
}

run();
```

Python

```
from pymongo import MongoClient

client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
db = client['test']
collection = db['videos']

result = list(collection.aggregate([
    {
        '$match': {
            '$isArray': '$inventory'
        }
    },
    {
        '$project': {
            '_id': 1,
            'name': 1,
            'inventory.location': 1,
            'inventory.count': 1
        }
    }
]))

print(result)

client.close()
```

$ ist Zahl

Neu ab Version 8.0.1.

Der `$isNumber` Operator in Amazon DocumentDB gibt einen booleschen Wert zurück, der angibt, ob ein bestimmter Ausdruck in einen numerischen Typ (Integer, Dezimal, Double, Long) aufgelöst wird. Beachten Sie, dass Arrays nicht als numerisch betrachtet werden, auch wenn alle Elemente im Array Zahlen sind. Wird beispielsweise auf den Wert `$isNumber` angewendet, der `[1, 2, 3]` zurückgegeben `false` wird.

Parameter

- `expression`: Jeder Ausdruck, um zu überprüfen, ob er in einen numerischen Typ aufgelöst wird.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$isNumber` Operator verwendet wird, um zu überprüfen, ob Feldwerte numerisch sind.

Erstellen Sie Beispieldokumente

```
db.data.insertMany([
  {_id: 1, value: 42},
  {_id: 2, value: "hello"},
  {_id: 3, value: 3.14},
  {_id: 4, value: null}
]);
```

Beispiel für eine Abfrage

```
db.data.aggregate([
  { $project: { isNum: { $isNumber: "$value" } } }
]);
```

Ausgabe

```
[
  {_id: 1, isNum: true},
  {_id: 2, isNum: false},
  {_id: 3, isNum: true},
  {_id: 4, isNum: false}
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$isNumber` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');
```

```
async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  try {
    await client.connect();
    const db = client.db('test');
    const collection = db.collection('data');
    const result = await collection.aggregate([
      { $project: { isNum: { $isNumber: "$value" } } }
    ]).toArray();
    console.log(result);
  } finally {
    await client.close();
  }
}
example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    try:
        db = client['test']
        collection = db['data']
        result = list(collection.aggregate([
            {'$project': {'isNum': {'$isNumber': '$value'}}}
        ]))
        print(result)
    finally:
        client.close()

example()
```

$ iso DayOfWeek

Der `$isoDayOfWeek` Operator in Amazon DocumentDB gibt den ISO-Wochentag für ein Datum als Ganzzahl zurück. Das ISO-Wochendatumssystem definiert jede Woche, die an einem Montag

beginnt und an einem Sonntag endet, wobei Woche 1 die Woche ist, die den ersten Donnerstag des Jahres enthält.

Parameter

- `expression`: Der Datumsausdruck, für den der ISO-Wochentag zurückgegeben werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$isoDayOfWeek` Operator verwendet wird, um den ISO-Wochentag für eine Reihe von Ereignisdokumenten abzurufen.

Erstellen Sie Beispieldokumente

```
db.events.insertMany([
  { _id: 1, eventDate: ISODate("2023-04-01T12:00:00Z") },
  { _id: 2, eventDate: ISODate("2023-04-02T12:00:00Z") },
  { _id: 3, eventDate: ISODate("2023-04-03T12:00:00Z") },
  { _id: 4, eventDate: ISODate("2023-04-04T12:00:00Z") },
  { _id: 5, eventDate: ISODate("2023-04-05T12:00:00Z") },
  { _id: 6, eventDate: ISODate("2023-04-06T12:00:00Z") },
  { _id: 7, eventDate: ISODate("2023-04-07T12:00:00Z") }
]);
```

Beispiel für eine Abfrage

```
db.events.aggregate([
  { $project: {
    _id: 1,
    eventDate: 1,
    isoDayOfWeek: { $isoDayOfWeek: "$eventDate" }
  }}
]);
```

Ausgabe

```
[
  { "_id": 1, "eventDate": ISODate("2023-04-01T12:00:00Z"), "isoDayOfWeek": 6 },
  { "_id": 2, "eventDate": ISODate("2023-04-02T12:00:00Z"), "isoDayOfWeek": 7 },
  { "_id": 3, "eventDate": ISODate("2023-04-03T12:00:00Z"), "isoDayOfWeek": 1 },
```

```
  { "_id": 4, "eventDate": ISODate("2023-04-04T12:00:00Z"), "isoDayOfWeek": 2 },
  { "_id": 5, "eventDate": ISODate("2023-04-05T12:00:00Z"), "isoDayOfWeek": 3 },
  { "_id": 6, "eventDate": ISODate("2023-04-06T12:00:00Z"), "isoDayOfWeek": 4 },
  { "_id": 7, "eventDate": ISODate("2023-04-07T12:00:00Z"), "isoDayOfWeek": 5 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$isoDayOfWeek` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const events = db.collection('events');

  const result = await events.aggregate([
    {
      $project: {
        _id: 1,
        eventDate: 1,
        isoDayOfWeek: { $isoDayOfWeek: '$eventDate' }
      }
    }
  ]).toArray();

  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
```

```
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    events = db.events

    result = list(events.aggregate([
        {
            '$project': {
                '_id': 1,
                'eventDate': 1,
                'isoDayOfWeek': { '$isoDayOfWeek': '$eventDate' }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$isoWeek

Der `$isoWeek` Operator in Amazon DocumentDB gibt die ISO-Wochennummer für ein Datum zurück. Das ISO-Wochendatumssystem ist eine Methode zur Nummerierung von Wochen in einem Jahr, wobei die erste Woche eines neuen Jahres die Woche ist, die den ersten Donnerstag des Jahres enthält. Dies unterscheidet sich vom gregorianischen Kalender, bei dem die erste Woche eines neuen Jahres die Woche ist, die den 1. Januar enthält.

Parameter

Keine

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$isoWeek` Operator verwendet wird, um die ISO-Wochennummer für ein bestimmtes Datum abzurufen.

Erstellen Sie Beispieldokumente

```
db.dates.insertMany([
  { _id: 1, date: new ISODate("2022-01-01") },
```

```
  { _id: 2, date: new ISODate("2022-12-31") },
  { _id: 3, date: new ISODate("2023-01-01") }
])
```

Beispiel für eine Abfrage

```
db.dates.aggregate([
  {
    $project: {
      _id: 1,
      isoWeek: { $isoWeek: "$date" }
    }
  }
])
```

Ausgabe

```
[
  { "_id": 1, "isoWeek": 52 },
  { "_id": 2, "isoWeek": 52 },
  { "_id": 3, "isoWeek": 1 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$isoWeek` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('dates');

  const result = await collection.aggregate([
    {
      $project: {
```

```
        _id: 1,
        isoWeek: { $isoWeek: "$date" }
      }
    }
  ]).toArray();

  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['dates']

    result = list(collection.aggregate([
        {
            '$project': {
                '_id': 1,
                'isoWeek': { '$isoWeek': '$date' }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$ iso WeekYear

Der `$isoWeekYear` Operator in Amazon DocumentDB gibt die ISO-8601-Wochenjahreszahl für ein bestimmtes Datum zurück. Die ISO-Wochenjahreszahl unterscheidet sich vom gregorianischen

Kalenderjahr dadurch, dass sich das Wochenjahr vom Kalenderjahr unterscheiden kann, insbesondere zu Beginn und Ende des Jahres.

Parameter

- `expression`: Der Datumsausdruck, für den die Jahreszahl der ISO-8601-Woche zurückgegeben werden soll.

Beispiel (MongoDB Shell)

Dieses Beispiel zeigt, wie der `$isoWeekYear` Operator verwendet wird, um das ISO 8601-Wochenjahr für das Datumsfeld jedes Dokuments in der Eventsammlung abzurufen.

Erstellen Sie Beispieldokumente

```
db.events.insertMany([
  { _id: 1, name: "Event 1", date: ISODate("2022-12-31T00:00:00Z") },
  { _id: 2, name: "Event 2", date: ISODate("2023-01-01T00:00:00Z") },
  { _id: 3, name: "Event 3", date: ISODate("2023-01-02T00:00:00Z") }
]);
```

Beispiel für eine Abfrage

```
db.events.aggregate([
  { $project: {
    name: 1,
    isoWeekYear: { $isoWeekYear: "$date" }
  }}
]);
```

Ausgabe

```
[
  { "_id" : 1, "name" : "Event 1", "isoWeekYear" : 2023 },
  { "_id" : 2, "name" : "Event 2", "isoWeekYear" : 2023 },
  { "_id" : 3, "name" : "Event 3", "isoWeekYear" : 2023 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$isoWeekYear` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const events = db.collection('events');

  const result = await events.aggregate([
    { $project: {
      name: 1,
      isoWeekYear: { $isoWeekYear: '$date' }
    }}
  ]).toArray();

  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    events = db.events

    result = list(events.aggregate([
        { '$project': {
            'name': 1,
```

```
            'isoWeekYear': { '$isoWeekYear': '$date' }
        }}
    ]))

    print(result)
    client.close()

example()
```

$last

Der `$last` Operator in Amazon DocumentDB wird verwendet, um das letzte Element in einem Array zurückzugeben, das den Abfragekriterien entspricht. Es ist besonders nützlich, um das neueste oder letzte Element in einem Array abzurufen, das eine bestimmte Bedingung erfüllt.

Parameter

- `expression`: Der Ausdruck, der den Array-Elementen entspricht.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung des `$last` Operators in Kombination mit `$filter` zum Abrufen des letzten Elements aus einem Array, das eine bestimmte Bedingung erfüllt (z. B. ist das Thema „Wissenschaft“).

Erstellen Sie Beispieldokumente

```
db.collection.insertMany([
  {
    "_id": 1,
    "name": "John",
    "scores": [
      { "subject": "math", "score": 82 },
      { "subject": "english", "score": 85 },
      { "subject": "science", "score": 90 }
    ]
  },
  {
    "_id": 2,
    "name": "Jane",
    "scores": [
```

```
      { "subject": "math", "score": 92 },
      { "subject": "english", "score": 88 },
      { "subject": "science", "score": 87 }
    ]
  },
  {
    "_id": 3,
    "name": "Bob",
    "scores": [
      { "subject": "math", "score": 75 },
      { "subject": "english", "score": 80 },
      { "subject": "science", "score": 85 }
    ]
  }
]);
```

Beispiel für eine Abfrage

```
db.collection.aggregate([
    { $match: { name: "John" } },
    {
      $project: {
        name: 1,
        lastScienceScore: {
          $last: {
            $filter: {
              input: "$scores",
              as: "score",
              cond: { $eq: ["$$score.subject", "science"] }
            }
          }
        }
      }
    }
  ]);
```

Ausgabe

```
[
  {
    _id: 1,
    name: 'John',
    lastScienceScore: { subject: 'science', score: 90 }
```

```
  }
]
```

Beispiel für die Verwendung von Fensteroperatoren (MongoDB Shell)

Neu ab Version 8.0.2.

Der `$last` Operator kann auch als Fensterbediener auf der `$setWindowFields` Bühne verwendet werden. In diesem Zusammenhang gibt es den Wert des `expression` letzten Dokuments in jedem Fenster zurück. Sie geben den Operator unter dem `output` Feld an und definieren optional die Fenstergrenzen mit einem `window` Dokument.

Erstellen Sie Beispieldokumente

```
db.stockPrices.insertMany([
  { _id: 1, ticker: "ABC", hour: 1, price: 50 },
  { _id: 2, ticker: "ABC", hour: 2, price: 45 },
  { _id: 3, ticker: "ABC", hour: 3, price: 60 },
  { _id: 4, ticker: "ABC", hour: 4, price: 40 },
  { _id: 5, ticker: "XYZ", hour: 1, price: 30 },
  { _id: 6, ticker: "XYZ", hour: 2, price: 35 },
  { _id: 7, ticker: "XYZ", hour: 3, price: 25 }
]);
```

Beispiel für eine Abfrage

Das folgende Beispiel partitioniert die Dokumente nach`ticker`, sortiert jede Partition nach und gibt den letzten (letzten) Preis zurück`hour`, der in jeder Partition aufgezeichnet wurde.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        lastPrice: {
          $last: "$price",
          window: { documents: ["unbounded", "unbounded"] }
        }
      }
    }
  }
```

```
]);
```

Ausgabe

```
[
  { "_id": 1, "ticker": "ABC", "hour": 1, "price": 50, "lastPrice": 40 },
  { "_id": 2, "ticker": "ABC", "hour": 2, "price": 45, "lastPrice": 40 },
  { "_id": 3, "ticker": "ABC", "hour": 3, "price": 60, "lastPrice": 40 },
  { "_id": 4, "ticker": "ABC", "hour": 4, "price": 40, "lastPrice": 40 },
  { "_id": 5, "ticker": "XYZ", "hour": 1, "price": 30, "lastPrice": 25 },
  { "_id": 6, "ticker": "XYZ", "hour": 2, "price": 35, "lastPrice": 25 },
  { "_id": 7, "ticker": "XYZ", "hour": 3, "price": 25, "lastPrice": 25 }
]
```

Jedes Dokument wird um den Preis des letzten Dokuments in seinem Fenster erweitert (das letzte Dokument `hour` in der Partition). `lastPrice`

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$last` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten. Die folgenden Beispiele zeigen sowohl die Verwendung von Ausdrücken (in`$project`) als auch die Verwendung von Fensteroperatoren (in`$setWindowFields`):

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  const db = client.db('test');

  // Expression usage: last element of a filtered array
  const collection = db.collection('collection');
  await collection.insertMany([
    {
      _id: 1,
      name: "John",
      scores: [
```

```
      { subject: "math", score: 82 },
      { subject: "english", score: 85 },
      { subject: "science", score: 90 }
    ]
  },
  {
    _id: 2,
    name: "Jane",
    scores: [
      { subject: "math", score: 92 },
      { subject: "english", score: 88 },
      { subject: "science", score: 87 }
    ]
  },
  {
    _id: 3,
    name: "Bob",
    scores: [
      { subject: "math", score: 75 },
      { subject: "english", score: 80 },
      { subject: "science", score: 85 }
    ]
  }
]);
const expressionResult = await collection.aggregate([
  { $match: { name: "John" } },
  {
    $project: {
      name: 1,
      lastScienceScore: {
        $last: {
          $filter: {
            input: "$scores",
            as: "score",
            cond: { $eq: ["$$score.subject", "science"] }
          }
        }
      }
    }
  }
]).toArray();
console.log('Expression result:', JSON.stringify(expressionResult, null, 2));

// Window operator usage: last value within each partition
```

```
  const stockPrices = db.collection('stockPrices');
  await stockPrices.insertMany([
    { _id: 1, ticker: "ABC", hour: 1, price: 50 },
    { _id: 2, ticker: "ABC", hour: 2, price: 45 },
    { _id: 3, ticker: "ABC", hour: 3, price: 60 },
    { _id: 4, ticker: "ABC", hour: 4, price: 40 },
    { _id: 5, ticker: "XYZ", hour: 1, price: 30 },
    { _id: 6, ticker: "XYZ", hour: 2, price: 35 },
    { _id: 7, ticker: "XYZ", hour: 3, price: 25 }
  ]);
  const windowResult = await stockPrices.aggregate([
    {
      $setWindowFields: {
        partitionBy: "$ticker",
        sortBy: { hour: 1 },
        output: {
          lastPrice: {
            $last: "$price",
            window: { documents: ["unbounded", "unbounded"] }
          }
        }
      }
    }
  ]).toArray();
  console.log('Window result:', JSON.stringify(windowResult, null, 2));

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():

    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    db = client.test
```

```
# Expression usage: last element of a filtered array
collection = db.collection
collection.insert_many([
    {
        "_id": 1,
        "name": "John",
        "scores": [
            { "subject": "math", "score": 82 },
            { "subject": "english", "score": 85 },
            { "subject": "science", "score": 90 }
        ]
    },
    {
        "_id": 2,
        "name": "Jane",
        "scores": [
            { "subject": "math", "score": 92 },
            { "subject": "english", "score": 88 },
            { "subject": "science", "score": 87 }
        ]
    },
    {
        "_id": 3,
        "name": "Bob",
        "scores": [
            { "subject": "math", "score": 75 },
            { "subject": "english", "score": 80 },
            { "subject": "science", "score": 85 }
        ]
    }
])
expression_pipeline = [
    { "$match": { "name": "John" } },
    {
        "$project": {
            "name": 1,
            "lastScienceScore": {
                "$last": {
                    "$filter": {
                        "input": "$scores",
                        "as": "score",
                        "cond": { "$eq": ["$$score.subject", "science"] }
                    }
                }
```

```
                }
            }
        }
    ]
    expression_result = list(collection.aggregate(expression_pipeline))
    print('Expression result:', expression_result)

    # Window operator usage: last value within each partition
    stock_prices = db.stockPrices
    stock_prices.insert_many([
        { "_id": 1, "ticker": "ABC", "hour": 1, "price": 50 },
        { "_id": 2, "ticker": "ABC", "hour": 2, "price": 45 },
        { "_id": 3, "ticker": "ABC", "hour": 3, "price": 60 },
        { "_id": 4, "ticker": "ABC", "hour": 4, "price": 40 },
        { "_id": 5, "ticker": "XYZ", "hour": 1, "price": 30 },
        { "_id": 6, "ticker": "XYZ", "hour": 2, "price": 35 },
        { "_id": 7, "ticker": "XYZ", "hour": 3, "price": 25 }
    ])
    window_pipeline = [
        {
            "$setWindowFields": {
                "partitionBy": "$ticker",
                "sortBy": { "hour": 1 },
                "output": {
                    "lastPrice": {
                        "$last": "$price",
                        "window": { "documents": ["unbounded", "unbounded"] }
                    }
                }
            }
        }
    ]
    window_result = list(stock_prices.aggregate(window_pipeline))
    print('Window result:', window_result)

    client.close()

example()
```

$ letztes N

Neu ab Version 8.0.1.

Der `$lastN` Operator in Amazon DocumentDB gibt die letzten N Elemente zurück. Wenn er als Akkumulator in einer `$group` Phase verwendet wird, gibt er ein Array der letzten N Werte in jeder Gruppe zurück. Wenn er als Array-Ausdrucksoperator verwendet wird, gibt er die letzten N Elemente eines Arrays zurück.

Parameter

- `input`: Der Ausdruck, der in das Feld oder Array aufgelöst wird, aus dem Werte zurückgegeben werden sollen.
- n: Eine positive Ganzzahl, die angibt, wie viele Werte zurückgegeben werden sollen. Bei Verwendung als Akkumulator in einer `$group` Phase n kann es sich auch um einen Ausdruck handeln, sofern er auf der Grundlage des `_id` Gruppenfeldes in eine positive Ganzzahl aufgelöst wird.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$lastN` Akkumulator verwendet wird, um die letzten beiden Mengen für jedes Element während der Aggregation abzurufen.

> **Note**
>
> `$lastN`wählt Werte in der Reihenfolge aus, in der Dokumente die `$group` Bühne erreichen. Um die letzten N Werte für eine bestimmte Reihenfolge zurückzugeben (z. B. nach Datum oder Punktzahl), fügen Sie eine `$sort` Stufe davor hinzu`$group`.

Erstellen Sie Beispieldokumente

```
db.sales.insertMany([
  { item: "abc", quantity: 10, date: ISODate("2023-01-01") },
  { item: "abc", quantity: 5, date: ISODate("2023-01-02") },
  { item: "abc", quantity: 8, date: ISODate("2023-01-03") },
  { item: "xyz", quantity: 15, date: ISODate("2023-01-01") },
  { item: "xyz", quantity: 7, date: ISODate("2023-01-02") },
  { item: "xyz", quantity: 3, date: ISODate("2023-01-03") }
]);
```

Beispiel für eine Abfrage

```
db.sales.aggregate([
  { $group: { _id: "$item", lastTwoQuantities: { $lastN: { input: "$quantity", n:
 2 } } } }
]);
```

Ausgabe

```
[
  { "_id": "abc", "lastTwoQuantities": [5, 8] },
  { "_id": "xyz", "lastTwoQuantities": [7, 3] }
]
```

Beispiel für die Verwendung von Ausdrücken (MongoDB Shell)

Der `$lastN` Operator kann auch als Ausdruck innerhalb einer `$project` Stufe verwendet werden, um die letzten N Elemente eines Array-Felds zurückzugeben.

Erstellen Sie Beispieldokumente

```
db.inventory.insertMany([
  { _id: 1, item: "abc", tags: ["red", "green", "blue", "yellow", "purple"] },
  { _id: 2, item: "xyz", tags: ["alpha", "beta", "gamma"] }
]);
```

Beispiel für eine Abfrage

```
db.inventory.aggregate([
  { $project: {
      lastThreeTags: { $lastN: { input: "$tags", n: 3 } }
    }}
]);
```

Ausgabe

```
[
  { "_id": 1, "lastThreeTags": ["blue", "yellow", "purple"] },
  { "_id": 2, "lastThreeTags": ["alpha", "beta", "gamma"] }
]
```

Beispiel für die Verwendung von Fensteroperatoren (MongoDB Shell)

Neu ab Version 8.0.2.

Der `$lastN` Operator kann auch als Fensterbediener auf der `$setWindowFields` Bühne verwendet werden. In diesem Zusammenhang gibt er ein Array mit bis zu den letzten n Werten für die Dokumente in jedem Fenster zurück. Sie geben den Operator unter dem `output` Feld an und definieren optional die Fenstergrenzen mit einem `window` Dokument.

> **Note**
>
> Bei Verwendung als Fensteroperator in `$setWindowFields` `$lastN` ist er auf 100 MB an Zwischendaten begrenzt. Eine Operation, die dieses Limit überschreitet, gibt einen Fehler zurück.

Erstellen Sie Beispieldokumente

```
db.stockPrices.insertMany([
  { _id: 1, ticker: "ABC", hour: 1, price: 50 },
  { _id: 2, ticker: "ABC", hour: 2, price: 45 },
  { _id: 3, ticker: "ABC", hour: 3, price: 60 },
  { _id: 4, ticker: "ABC", hour: 4, price: 40 },
  { _id: 5, ticker: "XYZ", hour: 1, price: 30 },
  { _id: 6, ticker: "XYZ", hour: 2, price: 35 },
  { _id: 7, ticker: "XYZ", hour: 3, price: 25 }
]);
```

Beispiel für eine Abfrage

Das folgende Beispiel partitioniert die Dokumente nach `tickerhour`, sortiert jede Partition nach und gibt die beiden aktuellsten Preise vom Anfang der Partition bis zum aktuellen Dokument zurück.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        lastTwoSoFar: {
          $lastN: { input: "$price", n: 2 },
          window: { documents: ["unbounded", "current"] }
```

```
        }
      }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "ticker": "ABC", "hour": 1, "price": 50, "lastTwoSoFar": [50] },
  { "_id": 2, "ticker": "ABC", "hour": 2, "price": 45, "lastTwoSoFar": [50, 45] },
  { "_id": 3, "ticker": "ABC", "hour": 3, "price": 60, "lastTwoSoFar": [45, 60] },
  { "_id": 4, "ticker": "ABC", "hour": 4, "price": 40, "lastTwoSoFar": [60, 40] },
  { "_id": 5, "ticker": "XYZ", "hour": 1, "price": 30, "lastTwoSoFar": [30] },
  { "_id": 6, "ticker": "XYZ", "hour": 2, "price": 35, "lastTwoSoFar": [30, 35] },
  { "_id": 7, "ticker": "XYZ", "hour": 3, "price": 25, "lastTwoSoFar": [35, 25] }
]
```

Jedes Dokument wird um die letzten beiden `price` Werte in seinem Fenster erweitert (in der `hour` Reihenfolge). `lastTwoSoFar`

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$lastN` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten. Die folgenden Beispiele zeigen die Verwendung von Akkumulatoren (in`$group`), Ausdrücken (in`$project`) und Fensteroperatoren (in`$setWindowFields`):

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  try {
    await client.connect();
    const db = client.db('test');

    // Accumulator usage: last N values per group
```

```
    const sales = db.collection('sales');
    await sales.insertMany([
      { item: "abc", quantity: 10, date: new Date("2023-01-01") },
      { item: "abc", quantity: 5, date: new Date("2023-01-02") },
      { item: "abc", quantity: 8, date: new Date("2023-01-03") },
      { item: "xyz", quantity: 15, date: new Date("2023-01-01") },
      { item: "xyz", quantity: 7, date: new Date("2023-01-02") },
      { item: "xyz", quantity: 3, date: new Date("2023-01-03") }
    ]);
    const accumulatorResult = await sales.aggregate([
      { $group: { _id: "$item", lastTwoQuantities: { $lastN: { input: "$quantity",
n: 2 } } } }
    ]).toArray();
    console.log('Accumulator result:', accumulatorResult);

    // Expression usage: last N elements of an array field
    const inventory = db.collection('inventory');
    await inventory.insertMany([
      { _id: 1, item: "abc", tags: ["red", "green", "blue", "yellow", "purple"] },
      { _id: 2, item: "xyz", tags: ["alpha", "beta", "gamma"] }
    ]);
    const expressionResult = await inventory.aggregate([
      { $project: { lastThreeTags: { $lastN: { input: "$tags", n: 3 } } } }
    ]).toArray();
    console.log('Expression result:', expressionResult);

    // Window operator usage: last N values over a window in each partition
    const stockPrices = db.collection('stockPrices');
    await stockPrices.insertMany([
      { _id: 1, ticker: "ABC", hour: 1, price: 50 },
      { _id: 2, ticker: "ABC", hour: 2, price: 45 },
      { _id: 3, ticker: "ABC", hour: 3, price: 60 },
      { _id: 4, ticker: "ABC", hour: 4, price: 40 },
      { _id: 5, ticker: "XYZ", hour: 1, price: 30 },
      { _id: 6, ticker: "XYZ", hour: 2, price: 35 },
      { _id: 7, ticker: "XYZ", hour: 3, price: 25 }
    ]);
    const windowResult = await stockPrices.aggregate([
      {
        $setWindowFields: {
          partitionBy: "$ticker",
          sortBy: { hour: 1 },
          output: {
            lastTwoSoFar: {
```

```
              $lastN: { input: "$price", n: 2 },
              window: { documents: ["unbounded", "current"] }
            }
          }
        }
      }
    ]).toArray();
    console.log('Window result:', windowResult);

  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient
from datetime import datetime

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    try:
        db = client['test']

        # Accumulator usage: last N values per group
        sales = db['sales']
        sales.insert_many([
            { 'item': 'abc', 'quantity': 10, 'date': datetime(2023, 1, 1) },
            { 'item': 'abc', 'quantity': 5, 'date': datetime(2023, 1, 2) },
            { 'item': 'abc', 'quantity': 8, 'date': datetime(2023, 1, 3) },
            { 'item': 'xyz', 'quantity': 15, 'date': datetime(2023, 1, 1) },
            { 'item': 'xyz', 'quantity': 7, 'date': datetime(2023, 1, 2) },
            { 'item': 'xyz', 'quantity': 3, 'date': datetime(2023, 1, 3) }
        ])
        accumulator_result = list(sales.aggregate([
            { '$group': { '_id': '$item', 'lastTwoQuantities': { '$lastN':
 { 'input': '$quantity', 'n': 2 } } } }
        ]))
```

```
        print('Accumulator result:', accumulator_result)

        # Expression usage: last N elements of an array field
        inventory = db['inventory']
        inventory.insert_many([
            { '_id': 1, 'item': 'abc', 'tags': ['red', 'green', 'blue', 'yellow',
'purple'] },
            { '_id': 2, 'item': 'xyz', 'tags': ['alpha', 'beta', 'gamma'] }
        ])
        expression_result = list(inventory.aggregate([
            { '$project': { 'lastThreeTags': { '$lastN': { 'input': '$tags', 'n':
3 } } } }
        ]))
        print('Expression result:', expression_result)

        # Window operator usage: last N values over a window in each partition
        stock_prices = db['stockPrices']
        stock_prices.insert_many([
            { '_id': 1, 'ticker': 'ABC', 'hour': 1, 'price': 50 },
            { '_id': 2, 'ticker': 'ABC', 'hour': 2, 'price': 45 },
            { '_id': 3, 'ticker': 'ABC', 'hour': 3, 'price': 60 },
            { '_id': 4, 'ticker': 'ABC', 'hour': 4, 'price': 40 },
            { '_id': 5, 'ticker': 'XYZ', 'hour': 1, 'price': 30 },
            { '_id': 6, 'ticker': 'XYZ', 'hour': 2, 'price': 35 },
            { '_id': 7, 'ticker': 'XYZ', 'hour': 3, 'price': 25 }
        ])
        window_result = list(stock_prices.aggregate([
            {
                '$setWindowFields': {
                    'partitionBy': '$ticker',
                    'sortBy': { 'hour': 1 },
                    'output': {
                        'lastTwoSoFar': {
                            '$lastN': { 'input': '$price', 'n': 2 },
                            'window': { 'documents': ['unbounded', 'current'] }
                        }
                    }
                }
            }
        ]))
        print('Window result:', window_result)

    finally:
        client.close()
```

```
example()
```

$let

Der `$let` Operator in Amazon DocumentDB wird verwendet, um Variablen an Werte zu binden und diese Variablen im Ausdruck zu verwenden. Es ermöglicht Ihnen, lokale Variablen zu definieren, die in nachfolgenden Ausdrücken innerhalb derselben Phase der Aggregationspipeline verwendet werden können.

Parameter

- `vars`: Ein Objekt, das die Variablen definiert, die im Ausdruck verwendet werden sollen.
- `in`: Der Ausdruck, in dem die im vars-Parameter definierten Variablen verwendet werden.

Beispiel (MongoDB Shell)

Dieses Beispiel demonstriert die Verwendung des `$let` Operators zur Berechnung der Fläche eines Rechtecks.

Erstellen Sie Beispieldokumente

```
db.shapes.insertMany([
  { name: "Rectangle 1", length: 5, width: 3 },
  { name: "Rectangle 2", length: 7, width: 4 },
  { name: "Rectangle 3", length: 6, width: 2 }
]);
```

Beispiel für eine Abfrage

```
db.shapes.aggregate([
  {
    $project: {
      name: 1,
      area: {
        $let: {
          vars: {
            length: "$length",
            width: "$width"
          },
```

```
          in: {
            $multiply: ["$$length", "$$width"]
          }
        }
      }
    }
  }
])
```

Ausgabe

```
[
  {
    "_id": ObjectId("6161e5b1a3eba3c7f2960d03"),
    "name": "Rectangle 1",
    "area": 15
  },
  {
    "_id": ObjectId("6161e5b1a3eba3c7f2960d04"),
    "name": "Rectangle 2",
    "area": 28
  },
  {
    "_id": ObjectId("6161e5b1a3eba3c7f2960d05"),
    "name": "Rectangle 3",
    "area": 12
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$let` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function calculateRectangleAreas() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
```

```
  const collection = db.collection('shapes');

  const result = await shapes.aggregate([
    {
      $project: {
        name: 1,
        area: {
          $let: {
            vars: {
              length: '$length',
              width: '$width'
            },
            in: {
              $multiply: ['$$length', '$$width']
            }
          }
        }
      }
    }
  ]).toArray();

  console.log(result);
  client.close();
}

calculateRectangleAreas();
```

Python

```
from pymongo import MongoClient

def calculate_rectangle_areas():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.shapes

    result = list(shapes.aggregate([
        {
            '$project': {
                'name': 1,
                'area': {
```

```
                    '$let': {
                        'vars': {
                            'length': '$length',
                            'width': '$width'
                        },
                        'in': {
                            '$multiply': ['$$length', '$$width']
                        }
                    }
                }
            }
        }
    ]))

    print(result)
    client.close()

calculate_rectangle_areas()
```

$limit

Der `$limit` Operator in Amazon DocumentDB wird verwendet, um die Anzahl der von einer Abfrage zurückgegebenen Dokumente einzuschränken. Er ähnelt dem `$limit` MongoDB-Operator, es gibt jedoch einige spezifische Überlegungen, wenn er mit Amazon DocumentDB verwendet wird.

In Amazon DocumentDB ist der `$limit` Operator nützlich für die Paginierung, bei der Sie eine Teilmenge der gesamten übereinstimmenden Dokumente abrufen möchten. Damit können Sie die Anzahl der Dokumente kontrollieren, die in jeder Antwort zurückgesendet werden, wodurch die Leistung verbessert und die Menge der über das Netzwerk übertragenen Daten reduziert wird.

Parameter

- `limit`: Die maximale Anzahl von Dokumenten, die zurückgegeben werden sollen. Dies muss ein nicht negativer Ganzzahlwert sein.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$limit` Operator verwendet wird, um maximal ein Dokument zurückzugeben, das dem angegebenen Filter entspricht.

Erstellen Sie Beispieldokumente

```
db.test.insertMany([
  { "_id": 1, "star_rating": 4, "comments": "apple is red" },
  { "_id": 2, "star_rating": 5, "comments": "comfortable couch" },
  { "_id": 3, "star_rating": 3, "comments": "apples, oranges - healthy fruit" },
  { "_id": 4, "star_rating": 5, "comments": "this is a great couch" },
  { "_id": 5, "star_rating": 5, "comments": "interesting couch" }
]);
```

Beispiel für eine Abfrage

```
db.test.createIndex({ comments: "text" });

db.test.find({
  $and: [
    { star_rating: 5 },
    { $text: { $search: "couch" } }
  ]
}).limit(1);
```

Ausgabe

```
[ { _id: 2, star_rating: 5, comments: 'comfortable couch' } ]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$limit` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function limitExample() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  try {
      await client.connect();

      const db = client.db('test');
```

```
      const collection = db.collection('test');

      await collection.createIndex({ comments: 'text' });

      const query = {
        $and: [
          { star_rating: 5 },
          { $text: { $search: "couch" } }
        ]
      };

      const result = await collection.find(query).limit(1).toArray();

      console.log(result);

    } catch (err) {
        console.error("Error:", err);
    } finally {
        await client.close();
    }

}

limitExample();
```

Python

```
from pymongo import MongoClient

def limit_example():
    try:
        client = MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

        db = client['test']
        collection = db['test']

        collection.create_index([('comments', 'text')])

        query = {
            '$and': [
                {'star_rating': 5},
```

```
                {'$text': {'$search': 'couch'}}
            ]
        }

        result = collection.find(query).limit(1)

        for doc in result:
            print(doc)

    except Exception as e:
        print(f"An error occurred: {e}")

    finally:
        client.close()

limit_example()
```

$ liste SearchIndexes

Neu ab Version 8.0.1.

Die `$listSearchIndexes` Aggregationsphase in Amazon DocumentDB gibt Informationen zu vorhandenen Suchindizes für eine Sammlung zurück. Sie muss die erste Phase in einer Aggregationspipeline sein.

Parameter

- `name`: (optional) Der Name des Suchindex, über den Informationen zurückgegeben werden sollen. Wenn er weggelassen wird, werden alle Suchindizes für die Sammlung zurückgegeben.

Ausgabefelder

Jedes zurückgegebene Dokument enthält die folgenden Felder:

- `name`: Der Name des Suchindex.
- `status`: Der Erstellungsstatus des Indexes. Einer von `READY` (erstellt und gültig), `BUILDING` (ein gleichzeitiger Build wird gerade erstellt) oder `FAILED` (die Indexerstellung wurde nicht erfolgreich abgeschlossen).
- `queryable`: Ein boolescher Wert, der angibt, ob der Index derzeit zur Bearbeitung von Abfragen verwendet werden kann. Dies gilt `false` für versteckte Indizes (die übrig bleibenREADY) und für

Indizes. `FAILED` Ein versteckter Index ist der Fall, wo er ist, `READY` aber `status` `queryable` ist. `false`

- `latestDefinitionVersion`: Ein Dokument, das `version` (die Version im Indexformat) und `createdAt` (die Uhrzeit, zu der der Index erstellt wurde) enthält.

Syntax

```
db.collection.aggregate([
  { $listSearchIndexes: {} }
])

// Or filter by name:
db.collection.aggregate([
  { $listSearchIndexes: { name: "mySearchIndex" } }
])
```

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$listSearchIndexes` Stagingbereich verwendet wird, um alle Suchindizes einer Sammlung aufzulisten.

Beispiel für eine Abfrage

```
db.movies.aggregate([
  { $listSearchIndexes: {} }
]);
```

Ausgabe

```
[
  {
    "name": "default",
    "status": "READY",
    "queryable": true,
    "latestDefinitionVersion": {
      "version": 2,
      "createdAt": ISODate("2026-05-11T21:12:57.974Z")
    }
  }
]
```

Ein ausgeblendeter Index bleibt bestehen`queryable: false`, meldet sich `READY` aber, da er nicht zur Bearbeitung von Abfragen verwendet werden kann, während er ausgeblendet ist:

```
[
  {
    "name": "myHiddenIndex",
    "status": "READY",
    "queryable": false,
    "latestDefinitionVersion": {
      "version": 2,
      "createdAt": ISODate("2026-05-11T21:12:57.974Z")
    }
  }
]
```

So filtern Sie nach einem bestimmten Indexnamen:

```
db.movies.aggregate([
  { $listSearchIndexes: { name: "default" } }
]);
```

Codebeispiele

Um ein Codebeispiel für die Verwendung der `$listSearchIndexes` Stufe anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  try {
    await client.connect();
    const db = client.db('test');
    const collection = db.collection('movies');

    // List all search indexes
    const allIndexes = await collection.aggregate([
```

```
      { $listSearchIndexes: {} }
    ]).toArray();
    console.log('All search indexes:', allIndexes);

    // List a specific search index by name
    const namedIndex = await collection.aggregate([
      { $listSearchIndexes: { name: "default" } }
    ]).toArray();
    console.log('Named index:', namedIndex);

  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    try:
        db = client['test']
        collection = db['movies']

        # List all search indexes
        all_indexes = list(collection.aggregate([
            { '$listSearchIndexes': {} }
        ]))
        print('All search indexes:', all_indexes)

        # List a specific search index by name
        named_index = list(collection.aggregate([
            { '$listSearchIndexes': { 'name': 'default' } }
        ]))
        print('Named index:', named_index)

    finally:
```

```
        client.close()

example()
```

$literal

Der `$literal` Operator in Amazon DocumentDB wird verwendet, um einen Literalwert innerhalb einer Aggregationspipeline-Phase darzustellen. Er ermöglicht es Ihnen, einen bestimmten Wert einzubeziehen, z. B. eine Zahl, eine Zeichenfolge oder einen booleschen Wert, ohne ihn als Feldverweis oder Ausdruck zu interpretieren.

Dieser Operator ist besonders nützlich, wenn Sie einen Literalwert als Teil einer komplexeren Aggregationspipeline einbeziehen müssen, z. B. beim Erstellen dynamischer Abfragefilter oder beim Durchführen von Berechnungen.

Parameter

Keine

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung des `$literal` Operators, um einen Literalwert in eine Aggregationspipeline aufzunehmen. Der `$literal` Operator wird verwendet, um den Wert 18 als Literalwert in den $gt-Ausdruck aufzunehmen. Auf diese Weise kann die Aggregationspipeline das Altersfeld mit dem Literalwert 18 vergleichen, um festzustellen, ob es sich bei der Person um einen Erwachsenen handelt.

Erstellen Sie Beispieldokumente

```
db.collection.insertMany([
  { "name": "John Doe", "age": 30, "city": "New York" },
  { "name": "Jane Doe", "age": 25, "city": "Los Angeles" },
  { "name": "Bob Smith", "age": 35, "city": "Chicago" }
]);
```

Beispiel für eine Abfrage

```
db.collection.aggregate([
  {
    $project: {
```

```
      name: 1,
      age: 1,
      city: 1,
      isAdult: { $gt: ["$age", { $literal: 18 }] }
    }
  }
]);
```

Ausgabe

```
[
  {
    "_id": ObjectId("601234567890abcdef012345"),
    "name": "John Doe",
    "age": 30,
    "city": "New York",
    "isAdult": true
  },
  {
    "_id": ObjectId("601234567890abcdef012346"),
    "name": "Jane Doe",
    "age": 25,
    "city": "Los Angeles",
    "isAdult": true
  },
  {
    "_id": ObjectId("601234567890abcdef012347"),
    "name": "Bob Smith",
    "age": 35,
    "city": "Chicago",
    "isAdult": true
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$literal` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');
```

```
async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('collection');

  const result = await collection.aggregate([
    {
      $project: {
        name: 1,
        age: 1,
        city: 1,
        isAdult: { $gt: ["$age", { $literal: 18 }] }
      }
    }
  ]).toArray();

  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.collection

    result = list(collection.aggregate([
        {
            '$project': {
                'name': 1,
                'age': 1,
                'city': 1,
                'isAdult': { '$gt': ["$age", { '$literal': 18 }] }
            }
```

```
        }
    ]))

    print(result)
    client.close()

example()
```

$ln

Neu ab Version 4.0.

Der `$ln` Operator in Amazon DocumentDB berechnet den natürlichen Logarithmus (Basis e) einer bestimmten Zahl. Er gibt den Logarithmus der Zahl zur Basis e zurück.

Parameter

- `expression`: Die Zahl, für die der natürliche Logarithmus berechnet wird.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung des `$log` Operators zur Berechnung des natürlichen Logarithmus einer Zahl.

Erstellen Sie Beispieldokumente

```
db.numbers.insertMany([
  { _id: 1, value: 10 },
  { _id: 2, value: 100 },
  { _id: 3, value: 1000 }
]);
```

Beispiel für eine Abfrage

```
db.numbers.aggregate([
  { $project: {
    _id: 1,
    naturalLog: { $ln: "$value" }
  }}
]);
```

Ausgabe

```
[
  { "_id" : 1, "naturalLog" : 2.302585092994046 },
  { "_id" : 2, "naturalLog" : 4.605170185988092 },
  { "_id" : 3, "naturalLog" : 6.907755278982137 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$ln` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  const db = client.db('test');
  const collection = db.collection('numbers');

  const result = await collection.aggregate([
    { $project: {
      _id: 1,
      naturalLog: { $ln: "$value" }
    }}
  ]).toArray();

  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
```

```
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    db = client.test
    collection = db.numbers

    result = list(collection.aggregate([
        { "$project": {
            "_id": 1,
            "naturalLog": { "$ln": "$value" }
        }}
    ]))

    print(result)
    client.close()

example()
```

$log

Neu ab Version 4.0.

Der `$log` Operator in Amazon DocumentDB berechnet den natürlichen Logarithmus einer Zahl. Er gibt den Logarithmus zur Basis E der angegebenen Zahl zurück.

Parameter

- `expression`: Die Zahl, für die der natürliche Logarithmus berechnet wird.
- `base`: Basiswert für die Log-Berechnung.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung des `$log` Operators zur Berechnung des natürlichen Logarithmus einer Zahl.

Erstellen Sie Beispieldokumente

```
db.numbers.insertMany([
  { _id: 1, value: 10 },
```

```
  { _id: 2, value: 100 },
  { _id: 3, value: 1000 }
]);
```

Beispiel für eine Abfrage

```
db.numbers.aggregate([
  { $project: {
    _id: 1,
    naturalLog: { $log: ["$value", 10] }
  }}
]);
```

Ausgabe

```
[
  { "_id" : 1, "naturalLog" : 1 },
  { "_id" : 2, "naturalLog" : 2 },
  { "_id" : 3, "naturalLog" : 2.9999999999999996 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$log` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  const db = client.db('test');
  const collection = db.collection('numbers');

  const result = await collection.aggregate([
    { $project: {
      _id: 1,
      naturalLog: { $log: ["$value", 10] }
```

```
    }}
  ]).toArray();

  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    db = client.test
    collection = db.numbers

    result = list(collection.aggregate([
        { "$project": {
            "_id": 1,
            "naturalLog": { "$log": ["$value", 10] }
        }}
    ]))

    print(result)
    client.close()

example()
```

$log10

Neu ab Version 4.0.

Der `$log10` Operator in Amazon DocumentDB wird verwendet, um den Logarithmus einer Zahl zur Basis 10 zu berechnen. Er ist nützlich, um logarithmische Berechnungen für numerische Felder innerhalb der Aggregationspipeline durchzuführen.

Parameter

- `expression`: Der numerische Ausdruck, für den der Logarithmus zur Basis 10 berechnet werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$log10` Operator verwendet wird, um den Logarithmus zur Basis 10 eines numerischen Feldes zu berechnen.

Erstellen Sie Beispieldokumente

```
db.numbers.insertMany([
  { _id: 1, value: 1 },
  { _id: 2, value: 10 },
  { _id: 3, value: 100 },
  { _id: 4, value: 1000 }
]);
```

Beispiel für eine Abfrage

```
db.numbers.aggregate([
  {
    $project: {
      _id: 1,
      log10Value: { $log10: "$value" }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "log10Value": 0 },
  { "_id": 2, "log10Value": 1 },
  { "_id": 3, "log10Value": 2 },
  { "_id": 4, "log10Value": 3 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$log10` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  let client;

  try {
    client = await MongoClient.connect('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

    const db = client.db('test');
    const collection = db.collection('numbers');

    const result = await collection.aggregate([
      {
        $project: {
          _id: 1,
          log10Value: { $log10: "$value" }
        }
      }
    ]).toArray();

    console.log(result);
  } catch (error) {
    console.error("An error occurred:", error);
  } finally {
    if (client) {
      await client.close();
    }
  }
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = None
    try:
```

```
        client = MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

        db = client.test
        collection = db.numbers

        result = list(collection.aggregate([
            {
                '$project': {
                    '_id': 1,
                    'log10Value': { '$log10': '$value' }
                }
            }
        ]))

        print(result)

    except Exception as e:
        print(f"An error occurred: {e}")

    finally:
        if client:
            client.close()

example()
```

$lt

Der `$lt` Aggregationsoperator vergleicht zwei Werte und gibt zurück, `true` wenn der erste Wert kleiner als der zweite ist, andernfalls kehrt `false` er zurück.

Parameter

- `expression1`: Der erste zu vergleichende Wert.
- `expression2`: Der zweite zu vergleichende Wert.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$lt` Operator verwendet wird, um Artikel mit geringem Lagerbestand zu identifizieren.

Erstellen Sie Beispieldokumente

```
db.warehouse.insertMany([
  { _id: 1, item: "Bolts", stock: 5 },
  { _id: 2, item: "Nuts", stock: 25 },
  { _id: 3, item: "Screws", stock: 8 }
]);
```

Beispiel für eine Abfrage

```
db.warehouse.aggregate([
  {
    $project: {
      item: 1,
      stock: 1,
      lowStock: { $lt: ["$stock", 10] }
    }
  }
]);
```

Ausgabe

```
[
  { _id: 1, item: 'Bolts', stock: 5, lowStock: true },
  { _id: 2, item: 'Nuts', stock: 25, lowStock: false },
  { _id: 3, item: 'Screws', stock: 8, lowStock: true }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$lt` Aggregationsoperators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
```

```
  const collection = db.collection('warehouse');

  const result = await collection.aggregate([
    {
      $project: {
        item: 1,
        stock: 1,
        lowStock: { $lt: ["$stock", 10] }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['warehouse']

    result = list(collection.aggregate([
        {
            '$project': {
                'item': 1,
                'stock': 1,
                'lowStock': { '$lt': ['$stock', 10] }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$lte

Der `$lte` Aggregationsoperator vergleicht zwei Werte und gibt zurück, `true` ob der erste Wert kleiner oder gleich dem zweiten ist, andernfalls kehrt `false` er zurück.

Parameter

- `expression1`: Der erste zu vergleichende Wert.
- `expression2`: Der zweite zu vergleichende Wert.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$lte` Operator verwendet wird, um budgetfreundliche Artikel zu identifizieren.

Erstellen Sie Beispieldokumente

```
db.menu.insertMany([
  { _id: 1, dish: "Salad", price: 8 },
  { _id: 2, dish: "Pasta", price: 12 },
  { _id: 3, dish: "Soup", price: 6 }
]);
```

Beispiel für eine Abfrage

```
db.menu.aggregate([
  {
    $project: {
      dish: 1,
      price: 1,
      affordable: { $lte: ["$price", 10] }
    }
  }
]);
```

Ausgabe

```
[
  { _id: 1, dish: 'Salad', price: 8, affordable: true },
  { _id: 2, dish: 'Pasta', price: 12, affordable: false },
  { _id: 3, dish: 'Soup', price: 6, affordable: true }
```

```
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$lte` Aggregationsoperators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('menu');

  const result = await collection.aggregate([
    {
      $project: {
        dish: 1,
        price: 1,
        affordable: { $lte: ["$price", 10] }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
```

```
    collection = db['menu']

    result = list(collection.aggregate([
        {
            '$project': {
                'dish': 1,
                'price': 1,
                'affordable': { '$lte': ['$price', 10] }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$lookup

Die `$lookup` Aggregationsphase in Amazon DocumentDB ermöglicht es Ihnen, eine linke äußere Verknüpfung zwischen zwei Sammlungen durchzuführen. Mit diesem Vorgang können Sie Daten aus mehreren Sammlungen auf der Grundlage übereinstimmender Feldwerte kombinieren. Dies ist besonders nützlich, wenn Sie Daten aus verwandten Sammlungen in Ihre Abfrageergebnisse einbeziehen müssen.

Parameter

- `from`: Der Name der Sammlung, mit der die Verknüpfung durchgeführt werden soll.
- `localField`: Das Feld aus den Eingabedokumenten, das mit dem abgeglichen werden sollforeignField.
- `foreignField`: Das Feld aus den Dokumenten in der `from` Sammlung, das mit dem abgeglichen werden solllocalField.
- `as`: Der Name des neuen Felds, das zu den Ausgabedokumenten hinzugefügt werden soll, die die passenden Dokumente aus der `from` Sammlung enthalten.
- `pipeline`: Optional, wenn `localField` und angegeben `foreignField` sind. Eine Aggregationspipeline, die auf der `from` Sammlung ausgeführt werden soll. Die Pipeline gibt Dokumente aus der `from` Sammlung zurück. Um alle Dokumente zurückzugeben, geben Sie eine leere Pipeline an`[]`. Die Pipeline darf die `$merge` Stufen `$out` oder nicht enthalten. Siehe Join-Bedingungen und korrelierte Unterabfragen (MongoDB Shell).

- `let`: Optional. Ein Dokument, das Variablen definiert, die in den `pipeline` Phasen verwendet werden sollen. Verwenden Sie die Variablenausdrücke, um in den Eingabedokumenten, die mit dem verknüpft sind, auf die Felder zuzugreifen`pipeline`. Verwenden Sie die `"$$<variable>"` Syntax, um in den Pipeline-Phasen auf eine Variable zu verweisen. Siehe Join-Bedingungen und korrelierte Unterabfragen (MongoDB Shell).

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt einen einfachen `$lookup` Vorgang, der Daten aus der `orders` Sammlung mit der `customers` Sammlung verknüpft.

Erstellen Sie Beispieldokumente

```
db.customers.insertMany([
  { _id: 1, name: "Alice" },
  { _id: 2, name: "Bob" },
  { _id: 3, name: "Charlie" }
]);

db.orders.insertMany([
  { _id: 1, customer_id: 1, total: 50 },
  { _id: 2, customer_id: 1, total: 100 },
  { _id: 3, customer_id: 2, total: 75 }
]);
```

Beispiel für eine Abfrage

```
db.customers.aggregate([
  {
    $lookup: {
      from: "orders",
      localField: "_id",
      foreignField: "customer_id",
      as: "orders"
    }
  }
]);
```

Ausgabe

```
[
```

```
  {
    _id: 1,
    name: 'Alice',
    orders: [
      { _id: 2, customer_id: 1, total: 100 },
      { _id: 1, customer_id: 1, total: 50 }
    ]
  },
  { _id: 3, name: 'Charlie', orders: [] },
  {
    _id: 2,
    name: 'Bob',
    orders: [ { _id: 3, customer_id: 2, total: 75 } ]
  }
]
```

Join-Bedingungen und korrelierte Unterabfragen (MongoDB Shell)

Neu ab Version 8.0.2.

Zusätzlich zu einer einzelnen Gleichheitsübereinstimmung kann `$lookup` in Amazon DocumentDB eine Aggregationspipeline für die verknüpfte Sammlung ausgeführt werden. Dies unterstützt unkorrelierte Unterabfragen, die dieselbe Unterabfrage für jedes Eingabedokument ausführen, und korrelierte Unterabfragen, bei denen Variablen verwendet werden, die aus den Feldern der einzelnen Eingabedokumente definiert wurden, um Ergebnisse zurückzugeben, die mit diesem Dokument korreliert sind.

Syntax

Um eine korrelierte Unterabfrage auszuführen, definieren Sie Variablen aus den Feldern des Eingabedokuments mit und referenzieren Sie sie in den folgenden Schritten: `let` `pipeline`

```
{
  $lookup: {
    from: <collection to join>,
    let: { <var_1>: <expression>, ..., <var_n>: <expression> },
    pipeline: [ <pipeline to run on the joined collection> ],
    as: <output array field>
  }
}
```

Parameter

- `let`: Ein Dokument, das Variablen aus den Feldern der Eingabedokumente definiert. Verwenden Sie die `"$$<variable>"` Syntax, um in den Pipeline-Phasen auf eine Variable zu verweisen. Auf die Variablen kann von den Stufen in der zugegriffen werden`pipeline`, einschließlich zusätzlicher `$lookup` Stufen, die in der `pipeline` verschachtelt sind. Eine `$match` Stufe erfordert die Verwendung des `$expr` Operators für den Zugriff auf die Variablen. Lassen Sie es aus`let`, um eine unkorrelierte Unterabfrage auszuführen.
- `pipeline`: Die Aggregationspipeline, die für die Sammlung ausgeführt werden soll. `from` Die Pipeline kann nicht direkt auf Felder der Eingabedokumente zugreifen. Definieren Sie stattdessen Variablen mit den Variablen `let` und verweisen Sie auf die Variablen. Um alle Dokumente zurückzugeben, geben Sie eine leere Pipeline an`[]`.

Die Operatoren `$eq`, `$gt`, `$gte`, `$lt`, und die `$lte` Vergleichsoperatoren, die in einem `$expr` Operator innerhalb einer `$match` Stufe platziert sind, können einen Index für die `from` Sammlung verwenden, wenn sie ein Feld der verknüpften Sammlung mit einer `let` Variablen oder einer Konstanten vergleichen.

Erstellen Sie Beispieldokumente

```
db.customers.insertMany([
  { _id: 1, name: "Alice", min_total: 60 },
  { _id: 2, name: "Bob", min_total: 50 },
  { _id: 3, name: "Charlie", min_total: 20 }
]);

db.orders.insertMany([
  { _id: 1, customer_id: 1, total: 50 },
  { _id: 2, customer_id: 1, total: 100 },
  { _id: 3, customer_id: 2, total: 75 }
]);
```

Beispiel für eine Abfrage

Das folgende Beispiel verknüpft alle Kunden mit ihren Bestellungen und behält nur die Bestellungen bei, deren `total` Wert größer oder gleich dem eigenen `min_total` Wert des Kunden ist. Die Join-Bedingung und der Filter pro Kunde enthalten beide `let` Referenzvariablen: `$expr`

```
db.customers.aggregate([
  {
    $lookup: {
```

```
      from: "orders",
      let: { customer_id: "$_id", min_total: "$min_total" },
      pipeline: [
        {
          $match: {
            $expr: {
              $and: [
                { $eq: [ "$customer_id", "$$customer_id" ] },
                { $gte: [ "$total", "$$min_total" ] }
              ]
            }
          }
        },
        { $project: { _id: 0, total: 1 } }
      ],
      as: "qualifying_orders"
    }
  }
]);
```

Ausgabe

```
[
  { _id: 1, name: 'Alice', min_total: 60, qualifying_orders: [ { total: 100 } ] },
  { _id: 2, name: 'Bob', min_total: 50, qualifying_orders: [ { total: 75 } ] },
  { _id: 3, name: 'Charlie', min_total: 20, qualifying_orders: [] }
]
```

Für jeden Kunden wird die Unterabfrage anhand der `min_total` and-Werte dieses Kunden ausgewertet, sodass jedes Dokument sein eigenes korreliertes Ergebnis erhält. `_id`

Korrelierte Unterabfragen mit übersichtlicher Syntax (MongoDB Shell)

Neu ab Version 8.0.2.

Amazon DocumentDB unterstützt auch die präzise korrelierte Unterabfragesyntax, die den Gleichheitsvergleich auf und mit einem kombiniert. `localField` `foreignField` `pipeline` Durch die präzise Syntax entfällt die Anforderung, die Gleichheitsübereinstimmung innerhalb eines Operators in einer `$expr` Stufe auszudrücken. `$match`

Syntax

```
{
```

```
  $lookup: {
    from: <collection to join>,
    localField: <field from the input documents>,
    foreignField: <field from the documents of the "from" collection>,
    let: { <var_1>: <expression>, ..., <var_n>: <expression> },
    pipeline: [ <pipeline to run> ],
    as: <output array field>
  }
}
```

Beispiel für eine Abfrage

Das folgende Beispiel gibt dieselben Ergebnisse wie das vorherige Beispiel für eine korrelierte Unterabfrage zurück. Die Gleichheitsübereinstimmung zwischen `_id` und `customer_id` wird mit `localField` und `foreignField` anstelle einer `$expr` Bedingung ausgedrückt:

```
db.customers.aggregate([
  {
    $lookup: {
      from: "orders",
      localField: "_id",
      foreignField: "customer_id",
      let: { min_total: "$min_total" },
      pipeline: [
        { $match: { $expr: { $gte: [ "$total", "$$min_total" ] } } },
        { $project: { _id: 0, total: 1 } }
      ],
      as: "qualifying_orders"
    }
  }
]);
```

Ausgabe

```
[
  { _id: 1, name: 'Alice', min_total: 60, qualifying_orders: [ { total: 100 } ] },
  { _id: 2, name: 'Bob', min_total: 50, qualifying_orders: [ { total: 75 } ] },
  { _id: 3, name: 'Charlie', min_total: 20, qualifying_orders: [] }
]
```

Unterstützte Stufen in korrelierten Unterabfragen

Wenn eine `$lookup` Pipeline verwendet`let`, können die folgenden Stufen auf die "`$$<variable>`" Variablen verweisen:

- `$match`(unter Verwendung des `$expr` Operators)
- `$project`
- `$addFields`
- `$set`
- `$group`
- `$bucket`
- `$bucketAuto`
- `$sortByCount`
- `$lookup`(verschachtelt)
- `$graphLookup`

Stufen, die Variablenausdrücke nicht auswerten, wie`$sort`,`$skip`,`$limit`, und `$count``$sample`, und `$unwind``$unset`, können ebenfalls in der Pipeline verwendet werden, können aber nicht auf die Variablen verweisen.

Die folgenden Stufen werden in einer `$lookup` Pipeline, die Folgendes verwendet `$facet``$redact`, nicht unterstütztlet: `$replaceRoot``$replaceWith`,`$geoNear`,`$setWindowFields`,`$fill`, und `$match` with`$text`.

Einschränkungen

- Die `$merge` Stufen `$out` oder `pipeline` können nicht enthalten.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$lookup` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');
```

```
async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  await client.connect();

  const db = client.db('test');

  const result = await db.collection('customers').aggregate([
    {
      $lookup: {
        from: 'orders',
        localField: '_id',
        foreignField: 'customer_id',
        as: 'orders'
      }
    }
  ]).toArray();

  console.log(JSON.stringify(result, null, 2));
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    db = client.test

    collection = db.customers

    pipeline = [
        {
            "$lookup": {
                "from": "orders",
```

```
                "localField": "_id",
                "foreignField": "customer_id",
                "as": "orders"
            }
        }
    ]

    result = collection.aggregate(pipeline)

    for doc in result:
        print(doc)

    client.close()

example()
```

Codebeispiele für korrelierte Unterabfragen

Um ein Codebeispiel für die Verwendung der `$lookup` Stufe mit einer korrelierten Unterabfrage (mithilfe von `let` und`pipeline`) anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  const db = client.db('test');

  const customers = db.collection('customers');
  await customers.insertMany([
    { _id: 1, name: "Alice", min_total: 60 },
    { _id: 2, name: "Bob", min_total: 50 },
    { _id: 3, name: "Charlie", min_total: 20 }
  ]);
  const orders = db.collection('orders');
  await orders.insertMany([
    { _id: 1, customer_id: 1, total: 50 },
```

```
    { _id: 2, customer_id: 1, total: 100 },
    { _id: 3, customer_id: 2, total: 75 }
  ]);

  // Correlated subquery: filter each customer's orders with a per-customer
 threshold
  const result = await customers.aggregate([
    {
      $lookup: {
        from: 'orders',
        localField: '_id',
        foreignField: 'customer_id',
        let: { min_total: '$min_total' },
        pipeline: [
          { $match: { $expr: { $gte: [ '$total', '$$min_total' ] } } },
          { $project: { _id: 0, total: 1 } }
        ],
        as: 'qualifying_orders'
      }
    }
  ]).toArray();

  console.log(JSON.stringify(result, null, 2));
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    db = client.test

    customers = db.customers
    customers.insert_many([
        { "_id": 1, "name": "Alice", "min_total": 60 },
        { "_id": 2, "name": "Bob", "min_total": 50 },
```

```
        { "_id": 3, "name": "Charlie", "min_total": 20 }
    ])
    orders = db.orders
    orders.insert_many([
        { "_id": 1, "customer_id": 1, "total": 50 },
        { "_id": 2, "customer_id": 1, "total": 100 },
        { "_id": 3, "customer_id": 2, "total": 75 }
    ])

    # Correlated subquery: filter each customer's orders with a per-customer
 threshold
    pipeline = [
        {
            "$lookup": {
                "from": "orders",
                "localField": "_id",
                "foreignField": "customer_id",
                "let": { "min_total": "$min_total" },
                "pipeline": [
                    { "$match": { "$expr": { "$gte": [ "$total", "$
$min_total" ] } } },
                    { "$project": { "_id": 0, "total": 1 } }
                ],
                "as": "qualifying_orders"
            }
        }
    ]

    result = customers.aggregate(pipeline)

    for doc in result:
        print(doc)

    client.close()

example()
```

$ ltrim

Neu ab Version 4.0.

Wird von Elastic Cluster nicht unterstützt.

Der `$ltrim` Operator in Amazon DocumentDB wird verwendet, um führende Zeichen aus einer Zeichenfolge zu entfernen. Standardmäßig entfernt er führende Leerzeichen, aber Sie können auch eine Reihe von Zeichen angeben, die entfernt werden sollen, indem Sie das Argument chars übergeben.

Parameter

- `input`: Die Eingabezeichenfolge, aus der führende Leerzeichen entfernt werden sollen.
- `chars`: (optional) Um bestimmte Zeichen zu entfernen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung von`$ltrim`, um bestimmte Zeichen (“ *“) am Anfang einer Zeichenfolge zu entfernen.

Erstellen Sie Beispieldokumente

```
db.collection.insertMany([
  { name: " *John Doe", age: 30 },
  { name: "Jane Doe*", age: 25 },
  { name: "  Bob Smith  ", age: 35 }
]);
```

Beispiel für eine Abfrage

```
db.collection.aggregate([
  {
    $project: {
      _id: 0,
      name: {
        $ltrim: { input: "$name", chars: " *" }
      },
      age: 1
    }
  }
]);
```

Ausgabe

```
[
  { "name": "John Doe", "age": 30 },
```

```
  { "name": "Jane Doe ", "age": 25 },
  { "name": "Bob Smith  ", "age": 35 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$ltrim` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  try {
    await client.connect();

    const db = client.db('test');
    const collection = db.collection('collection');

    const pipeline = [
      {
        $project: {
          _id: 0,
          name: {
            $ltrim: {
              input: '$name',
              chars: ' *'
            }
          },
          age: 1
        }
      }
    ];

    const result = await collection.aggregate(pipeline).toArray();

    console.dir(result, { depth: null });
```

```
  } finally {
    await client.close();
  }
}

example().catch(console.error);
```

Python

```
from pymongo import MongoClient

def example():
    try:
        client = MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

        db = client.test
        collection = db.collection

        pipeline = [
            {
                "$project": {
                    "_id": 0,
                    "name": {
                        "$ltrim": {
                            "input": "$name",
                            "chars": " *"
                        }
                    },
                    "age": 1
                }
            }
        ]

        results = collection.aggregate(pipeline)

        for doc in results:
            print(doc)

    except Exception as e:
        print(f"An error occurred: {e}")
```

```
    finally:
        client.close()

example()
```

$map

Der `$map` Operator in Amazon DocumentDB ermöglicht es Ihnen, einen bestimmten Ausdruck auf jedes Element in einem Array anzuwenden und ein neues Array mit den transformierten Elementen zurückzugeben. Dieser Operator ist besonders nützlich für die Manipulation und Transformation von Daten innerhalb von Arrays. Er kann dazu beitragen, Ihren Anwendungscode zu vereinfachen und die Abfrageleistung zu verbessern, indem die Array-Verarbeitung auf Datenbankebene verlagert wird.

Parameter

- `input`: Das Array, das transformiert werden soll.
- `as`: (optional) Der Name der Variablen, die innerhalb des in-Ausdrucks verwendet werden soll, um das aktuelle Element darzustellen, das gerade verarbeitet wird.
- `in`: Der Ausdruck, der auf jedes Element im Eingabearray angewendet werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der $map-Operator verwendet wird, um ein Zahlenarray zu transformieren, wobei jeder Wert verdoppelt wird.

Erstellen Sie Beispieldokumente

```
db.collection.insertMany([
  { _id: 1, numbers: [1, 2, 3, 4, 5] },
  { _id: 2, numbers: [10, 20, 30, 40, 50] }
])
```

Beispiel für eine Abfrage

```
db.collection.aggregate([
  {
    $project: {
      doubledNumbers: { $map: { input: "$numbers", as: "num", in: { $multiply: ["$
$num", 2] } } }
```

```
    }
  }
])
```

Ausgabe

```
[
  { _id: 1, doubledNumbers: [2, 4, 6, 8, 10] },
  { _id: 2, doubledNumbers: [20, 40, 60, 80, 100] }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des $map Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('collection');

  const result = await collection.aggregate([
    {
      $project: {
        doubledNumbers: { $map: { input: "$numbers", as: "num", in: { $multiply: ["$
$num", 2] } } }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.collection

    result = list(collection.aggregate([
        {
            '$project': {
                'doubledNumbers': { '$map': { 'input': '$numbers', 'as': 'num',
 'in': { '$multiply': ['$$num', 2] } } }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$match

Die `$match` Pipeline-Phase in Amazon DocumentDB wird verwendet, um die Eingabedokumente nur so zu filtern, dass sie den angegebenen Abfragekriterien entsprechen. Sie ist eine der am häufigsten verwendeten Pipeline-Phasen bei Aggregationsvorgängen. Die `$match` Phase wird vor allen anderen Pipeline-Phasen angewendet, sodass Sie die Anzahl der Dokumente, die in den nachfolgenden Phasen verarbeitet werden müssen, effizient reduzieren können.

Parameter

- `query`: Ein Dokument, das die Auswahlkriterien für die Operation zum Ausdruck bringt. Das Abfragedokument verwendet dieselbe Syntax wie die `find()` Methode.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung des `$match` Stagingbereichs zum Filtern von Dokumenten auf der Grundlage eines bestimmten Feldwerts.

Erstellen Sie Beispieldokumente

```
db.collection.insertMany([
  { _id: 1, name: "John", age: 25, city: "New York" },
  { _id: 2, name: "Jane", age: 30, city: "Los Angeles" },
  { _id: 3, name: "Bob", age: 35, city: "Chicago" },
  { _id: 4, name: "Alice", age: 40, city: "Miami" }
]);
```

Beispiel für eine Abfrage

```
db.collection.aggregate([
  { $match: { age: { $gt: 30 } } },
  { $project: { _id: 1, name: 1, city: 1 } }
]);
```

Ausgabe

```
[
  { "_id": 3, "name": "Bob", "city": "Chicago" },
  { "_id": 4, "name": "Alice", "city": "Miami" }
]
```

`$match`In der Phase werden die Dokumente so gefiltert, dass sie nur diejenigen enthalten, bei denen das `age` Feld größer als 30 ist.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$match` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
```

```
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  const db = client.db('test');
  const collection = db.collection('collection');

  const result = await collection.aggregate([
    { $match: { age: { $gt: 30 } } },
    { $project: { _id: 1, name: 1, city: 1 } }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    db = client['test']
    collection = db['collection']

    result = list(collection.aggregate([
        { '$match': { 'age': { '$gt': 30 } } },
        { '$project': { '_id': 1, 'name': 1, 'city': 1 } }
    ]))

    print(result)
    client.close()

example()
```

$max

Der `$max` Aggregationsoperator gibt den Maximalwert eines angegebenen Felds in einer Reihe von Dokumenten zurück. Dieser Operator ist nützlich, um den höchsten Wert in einer Reihe von Dokumenten zu finden.

Parameter

- `expression`: Der Ausdruck, der zur Berechnung des Maximalwerts verwendet werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$max` Operator verwendet wird, um die maximale Punktzahl in einer Sammlung von Studentendokumenten zu ermitteln. `$group`In der Phase werden alle Dokumente gruppiert, und der `$max` Operator wird verwendet, um den Maximalwert des `score` Felds für alle Dokumente zu berechnen.

Erstellen Sie Beispieldokumente

```
db.students.insertMany([
  { name: "John", score: 85 },
  { name: "Jane", score: 92 },
  { name: "Bob", score: 78 },
  { name: "Alice", score: 90 }
])
```

Beispiel für eine Abfrage

```
db.students.aggregate([
  { $group: { _id: null, maxScore: { $max: "$score" } } },
  { $project: { _id: 0, maxScore: 1 } }
])
```

Ausgabe

```
[ { maxScore: 92 } ]
```

Beispiel für die Verwendung von Fensteroperatoren (MongoDB Shell)

Neu ab Version 8.0.2.

Der `$max` Operator kann auch als Fensterbediener auf der `$setWindowFields` Bühne verwendet werden. In diesem Zusammenhang gibt es den Maximalwert für die Dokumente in jedem Fenster zurück. Sie geben den Operator unter dem `output` Feld an und definieren optional die Fenstergrenzen mit einem `window` Dokument.

> **Note**
>
> Bei Verwendung als Fensteroperator in `$setWindowFields` `$max` ist er auf 100 MB an Zwischendaten begrenzt. Eine Operation, die dieses Limit überschreitet, gibt einen Fehler zurück.

Erstellen Sie Beispieldokumente

```
db.stockPrices.insertMany([
  { _id: 1, ticker: "ABC", hour: 1, price: 50 },
  { _id: 2, ticker: "ABC", hour: 2, price: 45 },
  { _id: 3, ticker: "ABC", hour: 3, price: 60 },
  { _id: 4, ticker: "ABC", hour: 4, price: 40 },
  { _id: 5, ticker: "XYZ", hour: 1, price: 30 },
  { _id: 6, ticker: "XYZ", hour: 2, price: 35 },
  { _id: 7, ticker: "XYZ", hour: 3, price: 25 }
]);
```

Beispiel für eine Abfrage

Das folgende Beispiel partitioniert die Dokumente nach`ticker`, sortiert jede Partition nach und gibt den höchsten Preis zurück`hour`, der vom Anfang der Partition bis zum aktuellen Dokument angezeigt wurde.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        highestSoFar: {
          $max: "$price",
          window: { documents: ["unbounded", "current"] }
        }
      }
```

```
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "ticker": "ABC", "hour": 1, "price": 50, "highestSoFar": 50 },
  { "_id": 2, "ticker": "ABC", "hour": 2, "price": 45, "highestSoFar": 50 },
  { "_id": 3, "ticker": "ABC", "hour": 3, "price": 60, "highestSoFar": 60 },
  { "_id": 4, "ticker": "ABC", "hour": 4, "price": 40, "highestSoFar": 60 },
  { "_id": 5, "ticker": "XYZ", "hour": 1, "price": 30, "highestSoFar": 30 },
  { "_id": 6, "ticker": "XYZ", "hour": 2, "price": 35, "highestSoFar": 35 },
  { "_id": 7, "ticker": "XYZ", "hour": 3, "price": 25, "highestSoFar": 35 }
]
```

Jedes Dokument wird um `highestSoFar` das laufende Maximum `price` innerhalb seiner Partition bis einschließlich des aktuellen Dokuments erweitert.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$max` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten. Die folgenden Beispiele zeigen sowohl die Verwendung des Akkumulators (in`$group`) als auch die Verwendung von Fensteroperatoren (in`$setWindowFields`):

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');

  // Accumulator usage: maximum value across a group
  const students = db.collection('students');
  await students.insertMany([
    { name: "John", score: 85 },
    { name: "Jane", score: 92 },
    { name: "Bob", score: 78 },
```

```
    { name: "Alice", score: 90 }
  ]);
  const accumulatorResult = await students.aggregate([
    { $group: { _id: null, maxScore: { $max: "$score" } } }
  ]).toArray();
  console.log('Accumulator result:', accumulatorResult);

  // Window operator usage: running maximum within each partition
  const stockPrices = db.collection('stockPrices');
  await stockPrices.insertMany([
    { _id: 1, ticker: "ABC", hour: 1, price: 50 },
    { _id: 2, ticker: "ABC", hour: 2, price: 45 },
    { _id: 3, ticker: "ABC", hour: 3, price: 60 },
    { _id: 4, ticker: "ABC", hour: 4, price: 40 },
    { _id: 5, ticker: "XYZ", hour: 1, price: 30 },
    { _id: 6, ticker: "XYZ", hour: 2, price: 35 },
    { _id: 7, ticker: "XYZ", hour: 3, price: 25 }
  ]);
  const windowResult = await stockPrices.aggregate([
    {
      $setWindowFields: {
        partitionBy: "$ticker",
        sortBy: { hour: 1 },
        output: {
          highestSoFar: {
            $max: "$price",
            window: { documents: ["unbounded", "current"] }
          }
        }
      }
    }
  ]).toArray();
  console.log('Window result:', windowResult);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient
```

```
def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']

    # Accumulator usage: maximum value across a group
    students = db.students
    students.insert_many([
        { "name": "John", "score": 85 },
        { "name": "Jane", "score": 92 },
        { "name": "Bob", "score": 78 },
        { "name": "Alice", "score": 90 }
    ])
    accumulator_result = list(students.aggregate([
        { "$group": { "_id": None, "maxScore": { "$max": "$score" } } }
    ]))
    print('Accumulator result:', accumulator_result)

    # Window operator usage: running maximum within each partition
    stock_prices = db.stockPrices
    stock_prices.insert_many([
        { "_id": 1, "ticker": "ABC", "hour": 1, "price": 50 },
        { "_id": 2, "ticker": "ABC", "hour": 2, "price": 45 },
        { "_id": 3, "ticker": "ABC", "hour": 3, "price": 60 },
        { "_id": 4, "ticker": "ABC", "hour": 4, "price": 40 },
        { "_id": 5, "ticker": "XYZ", "hour": 1, "price": 30 },
        { "_id": 6, "ticker": "XYZ", "hour": 2, "price": 35 },
        { "_id": 7, "ticker": "XYZ", "hour": 3, "price": 25 }
    ])
    window_result = list(stock_prices.aggregate([
        {
            "$setWindowFields": {
                "partitionBy": "$ticker",
                "sortBy": { "hour": 1 },
                "output": {
                    "highestSoFar": {
                        "$max": "$price",
                        "window": { "documents": ["unbounded", "current"] }
                    }
                }
            }
        }
    ]))
```

```
    print('Window result:', window_result)

    client.close()

example()
```

$ MaxN

Neu ab Version 8.0.1.

Der `$maxN` Operator in Amazon DocumentDB gibt die N größten Werte zurück. Wenn er als Akkumulator in einer `$group` Stufe verwendet wird, gibt er ein Array mit den N Maximalwerten in jeder Gruppe zurück. Wenn er als Array-Ausdrucksoperator verwendet wird, gibt er die N maximalen Elemente eines Arrays zurück.

Parameter

- `input`: Ein Ausdruck, der in das Array oder Feld aufgelöst wird, aus dem die n Maximalwerte zurückgegeben werden sollen.
- n: Ein Ausdruck, der in eine positive Ganzzahl aufgelöst wird und angibt, wie viele Maximalwerte zurückgegeben werden sollen.

Behavior

Verhalten des Operators „Array-Ausdruck

- Sie können keinen Wert von n weniger als angeben`1`.
- `$maxN`filtert die im `input` Array gefundenen `null` Werte heraus.
- Wenn der angegebene n Wert größer oder gleich der Anzahl der Elemente im `input` Array ist, werden alle Elemente im `input` Array `$maxN` zurückgegeben.
- Wenn es in einen Wert `input` aufgelöst wird, der kein Array ist, tritt bei der Aggregationsoperation ein Fehler auf.
- Wenn es sowohl numerische als auch Zeichenkettenelemente `input` enthält, werden die Zeichenkettenelemente gemäß der BSON-Vergleichsreihenfolge vor den numerischen Elementen sortiert.

Verhalten des Akkumulators

- Bei Verwendung als Akkumulator n muss es sich um einen positiven Integralausdruck handeln, der entweder eine Konstante ist oder vom `_id` Wert für abhängt. `$group`
- `$maxN`filtert Nullwerte und fehlende Werte heraus.
- Wenn die Gruppe weniger als n Elemente enthält, werden alle Elemente in der Gruppe `$maxN` zurückgegeben.
- `$maxN`vergleicht Eingabedaten gemäß der BSON-Vergleichsreihenfolge, um den geeigneten Ausgabetyp zu ermitteln. Wenn die Eingabedaten mehrere Datentypen enthalten, ist der `$maxN` Ausgabetyp der höchste in der Vergleichsreihenfolge.

Reihenfolge der Ausgaben

`$maxN`gibt Werte in keiner bestimmten Sortierreihenfolge zurück. Wenn die Garantie einer bestimmten Sortierreihenfolge erforderlich ist, verwenden Sie `$topN` stattdessen die Option oder schließen Sie das Ergebnis damit `$sortArray` ein.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$maxN` Akkumulator verwendet wird, um die beiden höchsten Punktzahlen für jedes Subjekt abzurufen.

Erstellen Sie Beispieldokumente

```
db.scores.insertMany([
  { subject: "math", score: 85 },
  { subject: "math", score: 72 },
  { subject: "math", score: 93 },
  { subject: "math", score: 68 },
  { subject: "science", score: 90 },
  { subject: "science", score: 78 },
  { subject: "science", score: 65 },
  { subject: "science", score: 88 }
]);
```

Beispiel für eine Abfrage

```
db.scores.aggregate([
  { $group: { _id: "$subject", highestTwo: { $maxN: { input: "$score", n: 2 } } } }
]);
```

Ausgabe

```
[
  { "_id": "math", "highestTwo": [85, 93] },
  { "_id": "science", "highestTwo": [88, 90] }
]
```

Beispiel für die Verwendung von Ausdrücken (MongoDB Shell)

Der `$maxN` Operator kann auch als Ausdruck innerhalb einer `$project` Stufe verwendet werden, um die N größten Elemente aus einem Array-Feld zurückzugeben.

Erstellen Sie Beispieldokumente

```
db.readings.insertMany([
  { _id: 1, sensor: "A", values: [45, 12, 78, 3, 56] },
  { _id: 2, sensor: "B", values: [90, 23, 67, 11, 44] }
]);
```

Beispiel für eine Abfrage

```
db.readings.aggregate([
  { $project: {
      highestThree: { $maxN: { input: "$values", n: 3 } }
    }}
]);
```

Ausgabe

```
[
  { "_id": 1, "highestThree": [45, 56, 78] },
  { "_id": 2, "highestThree": [44, 90, 67] }
]
```

Beispiel für die Verwendung von Fensteroperatoren (MongoDB Shell)

Neu ab Version 8.0.2.

Der `$maxN` Operator kann auch als Fensterbediener auf der `$setWindowFields` Bühne verwendet werden. In diesem Zusammenhang gibt er ein Array mit bis zu den n größten Werten für die

Dokumente in jedem Fenster zurück. Sie geben den Operator unter dem `output` Feld an und definieren optional die Fenstergrenzen mit einem `window` Dokument.

> **Note**
>
> Bei Verwendung als Fensteroperator in `$setWindowFields` `$maxN` ist er auf 100 MB an Zwischendaten begrenzt. Eine Operation, die dieses Limit überschreitet, gibt einen Fehler zurück.

Erstellen Sie Beispieldokumente

```
db.stockPrices.insertMany([
  { _id: 1, ticker: "ABC", hour: 1, price: 50 },
  { _id: 2, ticker: "ABC", hour: 2, price: 45 },
  { _id: 3, ticker: "ABC", hour: 3, price: 60 },
  { _id: 4, ticker: "ABC", hour: 4, price: 40 },
  { _id: 5, ticker: "XYZ", hour: 1, price: 30 },
  { _id: 6, ticker: "XYZ", hour: 2, price: 35 },
  { _id: 7, ticker: "XYZ", hour: 3, price: 25 }
]);
```

Beispiel für eine Abfrage

Das folgende Beispiel partitioniert die Dokumente nach`ticker`, sortiert jede Partition nach `hour` und gibt die beiden höchsten Preise vom Anfang der Partition bis zum aktuellen Dokument zurück.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        highestTwoSoFar: {
          $maxN: { input: "$price", n: 2 },
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "ticker": "ABC", "hour": 1, "price": 50, "highestTwoSoFar": [50] },
  { "_id": 2, "ticker": "ABC", "hour": 2, "price": 45, "highestTwoSoFar": [50, 45] },
  { "_id": 3, "ticker": "ABC", "hour": 3, "price": 60, "highestTwoSoFar": [60, 50] },
  { "_id": 4, "ticker": "ABC", "hour": 4, "price": 40, "highestTwoSoFar": [60, 50] },
  { "_id": 5, "ticker": "XYZ", "hour": 1, "price": 30, "highestTwoSoFar": [30] },
  { "_id": 6, "ticker": "XYZ", "hour": 2, "price": 35, "highestTwoSoFar": [35, 30] },
  { "_id": 7, "ticker": "XYZ", "hour": 3, "price": 25, "highestTwoSoFar": [35, 30] }
]
```

Jedes Dokument wird um die beiden größten `price` Werte in seinem Fenster erweitert. `highestTwoSoFar` Wie im Abschnitt Reihenfolge der Ausgaben erwähnt, ist die Reihenfolge der Werte innerhalb des zurückgegebenen Arrays nicht garantiert.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$maxN` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten. Die folgenden Beispiele zeigen die Verwendung von Akkumulatoren (in`$group`), Ausdrücken (in`$project`) und Fensteroperatoren (in`$setWindowFields`):

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  try {
    await client.connect();
    const db = client.db('test');

    // Accumulator usage: N largest values per group
    const scores = db.collection('scores');
    await scores.insertMany([
      { subject: "math", score: 85 },
      { subject: "math", score: 72 },
      { subject: "math", score: 93 },
```

```
    { subject: "math", score: 68 },
    { subject: "science", score: 90 },
    { subject: "science", score: 78 },
    { subject: "science", score: 65 },
    { subject: "science", score: 88 }
  ]);
  const accumulatorResult = await scores.aggregate([
    { $group: { _id: "$subject", highestTwo: { $maxN: { input: "$score", n:
2 } } } }
  ]).toArray();
  console.log('Accumulator result:', accumulatorResult);

  // Expression usage: N largest elements from an array field
  const readings = db.collection('readings');
  await readings.insertMany([
    { _id: 1, sensor: "A", values: [45, 12, 78, 3, 56] },
    { _id: 2, sensor: "B", values: [90, 23, 67, 11, 44] }
  ]);
  const expressionResult = await readings.aggregate([
    { $project: { highestThree: { $maxN: { input: "$values", n: 3 } } } }
  ]).toArray();
  console.log('Expression result:', expressionResult);

  // Window operator usage: N largest values over a window in each partition
  const stockPrices = db.collection('stockPrices');
  await stockPrices.insertMany([
    { _id: 1, ticker: "ABC", hour: 1, price: 50 },
    { _id: 2, ticker: "ABC", hour: 2, price: 45 },
    { _id: 3, ticker: "ABC", hour: 3, price: 60 },
    { _id: 4, ticker: "ABC", hour: 4, price: 40 },
    { _id: 5, ticker: "XYZ", hour: 1, price: 30 },
    { _id: 6, ticker: "XYZ", hour: 2, price: 35 },
    { _id: 7, ticker: "XYZ", hour: 3, price: 25 }
  ]);
  const windowResult = await stockPrices.aggregate([
    {
      $setWindowFields: {
        partitionBy: "$ticker",
        sortBy: { hour: 1 },
        output: {
          highestTwoSoFar: {
            $maxN: { input: "$price", n: 2 },
            window: { documents: ["unbounded", "current"] }
          }
```

```
          }
        }
      }
    ]).toArray();
    console.log('Window result:', windowResult);

  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    try:
        db = client['test']

        # Accumulator usage: N largest values per group
        scores = db['scores']
        scores.insert_many([
            { 'subject': 'math', 'score': 85 },
            { 'subject': 'math', 'score': 72 },
            { 'subject': 'math', 'score': 93 },
            { 'subject': 'math', 'score': 68 },
            { 'subject': 'science', 'score': 90 },
            { 'subject': 'science', 'score': 78 },
            { 'subject': 'science', 'score': 65 },
            { 'subject': 'science', 'score': 88 }
        ])
        accumulator_result = list(scores.aggregate([
            { '$group': { '_id': '$subject', 'highestTwo': { '$maxN': { 'input':
 '$score', 'n': 2 } } } }
        ]))
        print('Accumulator result:', accumulator_result)
```

```
        # Expression usage: N largest elements from an array field
        readings = db['readings']
        readings.insert_many([
            { '_id': 1, 'sensor': 'A', 'values': [45, 12, 78, 3, 56] },
            { '_id': 2, 'sensor': 'B', 'values': [90, 23, 67, 11, 44] }
        ])
        expression_result = list(readings.aggregate([
            { '$project': { 'highestThree': { '$maxN': { 'input': '$values', 'n':
 3 } } } }
        ]))
        print('Expression result:', expression_result)

        # Window operator usage: N largest values over a window in each partition
        stock_prices = db['stockPrices']
        stock_prices.insert_many([
            { '_id': 1, 'ticker': 'ABC', 'hour': 1, 'price': 50 },
            { '_id': 2, 'ticker': 'ABC', 'hour': 2, 'price': 45 },
            { '_id': 3, 'ticker': 'ABC', 'hour': 3, 'price': 60 },
            { '_id': 4, 'ticker': 'ABC', 'hour': 4, 'price': 40 },
            { '_id': 5, 'ticker': 'XYZ', 'hour': 1, 'price': 30 },
            { '_id': 6, 'ticker': 'XYZ', 'hour': 2, 'price': 35 },
            { '_id': 7, 'ticker': 'XYZ', 'hour': 3, 'price': 25 }
        ])
        window_result = list(stock_prices.aggregate([
            {
                '$setWindowFields': {
                    'partitionBy': '$ticker',
                    'sortBy': { 'hour': 1 },
                    'output': {
                        'highestTwoSoFar': {
                            '$maxN': { 'input': '$price', 'n': 2 },
                            'window': { 'documents': ['unbounded', 'current'] }
                        }
                    }
                }
            }
        ]))
        print('Window result:', window_result)

    finally:
        client.close()

example()
```

$ Median

Neu ab Version 8.0.1.

Der `$median` Operator in Amazon DocumentDB berechnet den Medianwert numerischer Daten. Als Akkumulator berechnet er in der Phase einer Aggregationspipeline den Median der numerischen Werte aller Dokumente innerhalb einer Gruppe. `$group` Als Ausdruck berechnet er den Median einer Reihe von Zahlen.

Parameter

- `input`: Ein Ausdruck, der in einen numerischen Wert oder ein Array numerischer Werte aufgelöst wird.
- `method`: Eine Zeichenfolge, die die Berechnungsmethode angibt. Derzeit `"approximate"` wird nur unterstützt, wobei der t-digest-Algorithmus verwendet wird.

Behavior

Die `"approximate"` Methode verwendet den T-Digest-Algorithmus, um einen ungefähren Median zu berechnen. Das Ergebnis ist ein vorhandener Wert aus dem Datensatz und nicht eine Interpolation zwischen Werten. Die Genauigkeit verbessert sich, wenn die Anzahl der Datenpunkte zunimmt.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$median` Operator verwendet wird, um den Medianwert des Testergebnisses pro Klasse zu berechnen.

Erstellen Sie Beispieldokumente

```
db.students.insertMany([
  { class: "A", score: 72 },
  { class: "A", score: 85 },
  { class: "A", score: 90 },
  { class: "A", score: 68 },
  { class: "A", score: 95 },
  { class: "B", score: 80 },
  { class: "B", score: 75 },
  { class: "B", score: 92 },
  { class: "B", score: 88 },
  { class: "B", score: 70 }
]);
```

Beispiel für eine Abfrage

```
db.students.aggregate([
  { $group: {
      _id: "$class",
      medianScore: { $median: { input: "$score", method: "approximate" } }
    }}
]);
```

Ausgabe

```
[
  { "_id": "A", "medianScore": 85 },
  { "_id": "B", "medianScore": 80 }
]
```

Beispiel für die Verwendung von Ausdrücken (MongoDB Shell)

Der `$median` Operator kann auch als Ausdruck innerhalb einer `$project` Stufe verwendet werden, um den Median eines Array-Feldes zu berechnen.

Erstellen Sie Beispieldokumente

```
db.surveys.insertMany([
  { _id: 1, ratings: [3, 5, 7, 9, 2] },
  { _id: 2, ratings: [10, 20, 30, 40, 50] },
  { _id: 3, ratings: [1, 1, 2, 3, 5] }
]);
```

Beispiel für eine Abfrage

```
db.surveys.aggregate([
  { $project: {
      medianRating: { $median: { input: "$ratings", method: "approximate" } }
    }}
]);
```

Ausgabe

```
[
  { "_id": 1, "medianRating": 5 },
  { "_id": 2, "medianRating": 30 },
```

```
  { "_id": 3, "medianRating": 2 }
]
```

Beispiel für die Verwendung von Fensteroperatoren (MongoDB Shell)

Neu ab Version 8.0.2.

Der `$median` Operator kann auch als Fensterbediener auf der `$setWindowFields` Bühne verwendet werden. In diesem Zusammenhang berechnet er den Median der numerischen Werte für die Dokumente in jedem Fenster. Sie geben den Operator unter dem `output` Feld an und definieren optional die Fenstergrenzen mit einem `window` Dokument.

> **Note**
>
> Bei Verwendung als Fensteroperator in `$setWindowFields` `$median` ist er auf 100 MB an Zwischendaten begrenzt. Eine Operation, die dieses Limit überschreitet, gibt einen Fehler zurück.

Erstellen Sie Beispieldokumente

```
db.temperatures.insertMany([
  { _id: 1, city: "A", reading: 60 },
  { _id: 2, city: "A", reading: 65 },
  { _id: 3, city: "A", reading: 70 },
  { _id: 4, city: "B", reading: 80 },
  { _id: 5, city: "B", reading: 85 },
  { _id: 6, city: "B", reading: 90 }
]);
```

Beispiel für eine Abfrage

Im folgenden Beispiel werden die Dokumente nach partitioniert `city` und der Median für alle `reading` Dokumente in jeder Partition berechnet.

```
db.temperatures.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$city",
      output: {
        medianReading: {
```

```
          $median: { input: "$reading", method: "approximate" },
          window: { documents: ["unbounded", "unbounded"] }
        }
      }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "city": "A", "reading": 60, "medianReading": 65 },
  { "_id": 2, "city": "A", "reading": 65, "medianReading": 65 },
  { "_id": 3, "city": "A", "reading": 70, "medianReading": 65 },
  { "_id": 4, "city": "B", "reading": 80, "medianReading": 85 },
  { "_id": 5, "city": "B", "reading": 85, "medianReading": 85 },
  { "_id": 6, "city": "B", "reading": 90, "medianReading": 85 }
]
```

Jedes Dokument wird um den ungefähren Medianwert aller Dokumente in `reading` seiner Partition erweitert. `medianReading`

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$median` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten. Die folgenden Beispiele zeigen die Verwendung von Akkumulatoren (in`$group`), Ausdrücken (in`$project`) und Fensteroperatoren (in`$setWindowFields`):

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const uri = 'mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false';
  const client = new MongoClient(uri);

  try {
    await client.connect();
    const db = client.db('test');
```

```
// Accumulator usage: median across grouped documents
const students = db.collection('students');
await students.insertMany([
  { class: "A", score: 72 },
  { class: "A", score: 85 },
  { class: "A", score: 90 },
  { class: "A", score: 68 },
  { class: "A", score: 95 },
  { class: "B", score: 80 },
  { class: "B", score: 75 },
  { class: "B", score: 92 },
  { class: "B", score: 88 },
  { class: "B", score: 70 }
]);
const accumulatorResult = await students.aggregate([
  { $group: {
      _id: "$class",
      medianScore: { $median: { input: "$score", method: "approximate" } }
    }}
]).toArray();
console.log('Accumulator result:', accumulatorResult);

// Expression usage: median of an array field
const surveys = db.collection('surveys');
await surveys.insertMany([
  { _id: 1, ratings: [3, 5, 7, 9, 2] },
  { _id: 2, ratings: [10, 20, 30, 40, 50] },
  { _id: 3, ratings: [1, 1, 2, 3, 5] }
]);
const expressionResult = await surveys.aggregate([
  { $project: {
      medianRating: { $median: { input: "$ratings", method: "approximate" } }
    }}
]).toArray();
console.log('Expression result:', expressionResult);

// Window operator usage: median across each partition
const temperatures = db.collection('temperatures');
await temperatures.insertMany([
  { _id: 1, city: "A", reading: 60 },
  { _id: 2, city: "A", reading: 65 },
  { _id: 3, city: "A", reading: 70 },
  { _id: 4, city: "B", reading: 80 },
```

```
      { _id: 5, city: "B", reading: 85 },
      { _id: 6, city: "B", reading: 90 }
    ]);
    const windowResult = await temperatures.aggregate([
      {
        $setWindowFields: {
          partitionBy: "$city",
          output: {
            medianReading: {
              $median: { input: "$reading", method: "approximate" },
              window: { documents: ["unbounded", "unbounded"] }
            }
          }
        }
      }
    ]).toArray();
    console.log('Window result:', windowResult);

  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    try:
        db = client['test']

        # Accumulator usage: median across grouped documents
        students = db['students']
        students.insert_many([
            { 'class': 'A', 'score': 72 },
            { 'class': 'A', 'score': 85 },
            { 'class': 'A', 'score': 90 },
```

```
            { 'class': 'A', 'score': 68 },
            { 'class': 'A', 'score': 95 },
            { 'class': 'B', 'score': 80 },
            { 'class': 'B', 'score': 75 },
            { 'class': 'B', 'score': 92 },
            { 'class': 'B', 'score': 88 },
            { 'class': 'B', 'score': 70 }
        ])
        accumulator_result = list(students.aggregate([
            { '$group': {
                '_id': '$class',
                'medianScore': { '$median': { 'input': '$score', 'method':
'approximate' } }
            }}
        ]))
        print('Accumulator result:', accumulator_result)

        # Expression usage: median of an array field
        surveys = db['surveys']
        surveys.insert_many([
            { '_id': 1, 'ratings': [3, 5, 7, 9, 2] },
            { '_id': 2, 'ratings': [10, 20, 30, 40, 50] },
            { '_id': 3, 'ratings': [1, 1, 2, 3, 5] }
        ])
        expression_result = list(surveys.aggregate([
            { '$project': {
                'medianRating': { '$median': { 'input': '$ratings', 'method':
'approximate' } }
            }}
        ]))
        print('Expression result:', expression_result)

        # Window operator usage: median across each partition
        temperatures = db['temperatures']
        temperatures.insert_many([
            { '_id': 1, 'city': 'A', 'reading': 60 },
            { '_id': 2, 'city': 'A', 'reading': 65 },
            { '_id': 3, 'city': 'A', 'reading': 70 },
            { '_id': 4, 'city': 'B', 'reading': 80 },
            { '_id': 5, 'city': 'B', 'reading': 85 },
            { '_id': 6, 'city': 'B', 'reading': 90 }
        ])
        window_result = list(temperatures.aggregate([
            {
```

```
                '$setWindowFields': {
                    'partitionBy': '$city',
                    'output': {
                        'medianReading': {
                            '$median': { 'input': '$reading', 'method':
 'approximate' },
                            'window': { 'documents': ['unbounded', 'unbounded'] }
                        }
                    }
                }
            }
        ]))
        print('Window result:', window_result)

    finally:
        client.close()

example()
```

$meta

Der `$meta` Aggregationsoperator greift auf Metadaten zu, die Dokumenten in einer Aggregationspipeline zugeordnet sind. Er wird häufig verwendet, um Ergebnisse der Textsuche abzurufen und Ergebnisse nach Relevanz zu sortieren.

Parameter

- `textScore`: Ruft das Ergebnis der Textsuche ab, das die Relevanz des Dokuments für die Suchabfrage angibt.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung des `$meta` Operators in einer Aggregationspipeline zum Abrufen und Sortieren nach Textsuchergebnissen.

Erstellen Sie Beispieldokumente

```
db.articles.createIndex({ content: "text" });

db.articles.insertMany([
```

```
  { _id: 1, title: "Python Programming", content: "Python is a versatile programming
 language used for web development." },
  { _id: 2, title: "Python Guide", content: "Learn Python programming with Python
 tutorials and Python examples." },
  { _id: 3, title: "Java Basics", content: "Java is another popular programming
 language." }
]);
```

Beispiel für eine Abfrage

```
db.articles.aggregate([
  { $match: { $text: { $search: "Python" } } },
  { $addFields: { score: { $meta: "textScore" } } },
  { $sort: { score: -1 } }
]);
```

Ausgabe

```
[
  {
    _id: 2,
    title: 'Python Guide',
    content: 'Learn Python programming with Python tutorials and Python examples.',
    score: 1.5
  },
  {
    _id: 1,
    title: 'Python Programming',
    content: 'Python is a versatile programming language used for web development.',
    score: 0.75
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$meta` Aggregationsoperators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');
```

```
async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('articles');

  const result = await collection.aggregate([
    { $match: { $text: { $search: "Python" } } },
    { $addFields: { score: { $meta: "textScore" } } },
    { $sort: { score: -1 } }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['articles']

    result = list(collection.aggregate([
        { '$match': { '$text': { '$search': 'Python' } } },
        { '$addFields': { 'score': { '$meta': 'textScore' } } },
        { '$sort': { 'score': -1 } }
    ]))

    print(result)
    client.close()

example()
```

$ zusammenführen

In 8.0 eingeführt

Die `$merge` Aggregationsphase in Amazon DocumentDB wird verwendet, um die Ergebnisse der vorherigen Pipeline-Phase in einer Zielsammlung zusammenzuführen. Dies ist nützlich, um Dokumente auf der Grundlage der Daten aus den Eingabedokumenten zu aktualisieren oder in eine Zielsammlung einzufügen.

In dieser `$merge` Phase können Sie auf der Grundlage der Übereinstimmungsbedingung zwischen den Eingabedokumenten und der Zielsammlung verschiedene Aktionen ausführen, z. B.:

```
- Insert new documents
- Update existing documents
- Delete documents
- Fail the operation if there are any conflicts
```

Parameter

- `into`: (erforderlich) Der Name der Zielsammlung, in der die Eingabedokumente zusammengeführt werden sollen.
- on: (erforderlich) Die Felder, die als Übereinstimmungsbedingung zwischen den Eingabedokumenten und der Zielsammlung verwendet werden sollen.
- `whenMatched`: (optional) Die Aktion, die ausgeführt werden soll, wenn das Eingabedokument mit einem vorhandenen Dokument in der Zielsammlung übereinstimmt. Unterstützte Werte sind `"merge"`, `"replace"`, `"keepExisting"` und `"fail"`. Amazon DocumentDB unterstützt nicht die Angabe einer Aggregationspipeline als Wert. `whenMatched`
- `whenNotMatched`: (optional) Die Aktion, die ausgeführt werden soll, wenn das Eingabedokument keinem Dokument in der Zielsammlung entspricht. Unterstützte Werte sind: `"insert"` und`"fail"`.
- `bypassDocumentValidation`: (optional) Wenn auf gesetzt, umgehen Dokumente`true`, die von der `$merge` Stufe in die Zielsammlung geschrieben werden, alle für diese Sammlung definierten Schemaüberprüfungsregeln (`$jsonSchema`). Die Standardeinstellung ist`false`, wodurch die Validierungsregeln der Zielsammlung durchgesetzt werden. Dies wird in Amazon DocumentDB 8.0 unterstützt.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$merge` Stagingbereich verwendet wird, um eine `users` Sammlung mit neuen Daten aus einer Eingabepipeline zu aktualisieren.

Erstellen Sie Beispieldokumente

```
db.users.insertMany([
  { _id: 1, name: "John Doe", email: "john@example.com" },
  { _id: 2, name: "Jane Smith", email: "jane@example.com" }
]);

db.inputData.insertMany([
  { _id: 1, name: "John Doe", email: "john@example.com", phone: "123-456-7890" },
  { _id: 3, name: "Bob Johnson", email: "bob@example.com", phone: "987-654-3210" }
]);
```

Beispiel für eine Abfrage

```
db.inputData.aggregate([
  {
    $merge: {
      into: "users",
      on: "_id",
      whenMatched: "merge",
      whenNotMatched: "insert"
    }
  }
])
```

Ausgabe

Nach dem Ausführen der `$merge` Pipeline wird die `users` Sammlung die folgenden Dokumente enthalten:

```
[
  { _id: 1, name: "John Doe", email: "john@example.com", phone: "123-456-7890" },
  { _id: 2, name: "Jane Smith", email: "jane@example.com" },
  { _id: 3, name: "Bob Johnson", email: "bob@example.com", phone: "987-654-3210" }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$merge` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

Hier ist ein Beispiel für die Verwendung des $merge-Operators in einer Node.js Anwendung:

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');

  await db.collection('inputData').aggregate([
    {
      $merge: {
        into: 'users',
        on: '_id',
        whenMatched: 'merge',
        whenNotMatched: 'insert'
      }
    }
  ]).toArray();

  const users = await db.collection('users').find({}).toArray();
  console.log(users);

  await client.close();
}

example();
```

Python

Hier ist ein Beispiel für die Verwendung des $merge-Operators in einer Python-Anwendung:

```
from pymongo import MongoClient

def example():
```

```
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']

    # Assumes collections 'users' and 'inputData' already exist with sample data
    db.inputData.aggregate([
        {
            '$merge': {
                'into': 'users',
                'on': '_id',
                'whenMatched': 'merge',
                'whenNotMatched': 'insert'
            }
        }
    ])

    users = list(db.users.find({}))
    print(users)

    client.close()

example()
```

$mergeObjects

Der `$mergeObjects` Operator in Amazon DocumentDB wird verwendet, um mehrere Dokumente oder Objekte zu einem einzigen Dokument zu kombinieren. Dieser Operator ist besonders nützlich, wenn Sie den Inhalt von zwei oder mehr Dokumenten oder Objekten zusammenführen müssen und dabei möglicherweise Werte aus einem Objekt mit denen eines anderen überschreiben müssen.

Parameter

- `expression1`: Das erste Objekt, das zusammengeführt werden soll.
- `expression2`: (optional) Das zweite Objekt, das zusammengeführt werden soll.
- `expression3`: (optional) Zusätzliche Objekte, die zusammengeführt werden sollen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$mergeObjects` Operator verwendet wird, um zwei Objekte zu kombinieren.

Erstellen Sie Beispieldokumente

```
db.collection.insertMany([
  { "_id": 1, "name": "John", "address": { "city": "New York", "state": "NY" } },
  { "_id": 2, "name": "Jane", "address": { "city": "Los Angeles", "state": "CA" } }
]);
```

Beispiel für eine Abfrage

```
db.collection.aggregate([
  {
    $project: {
      "combinedAddress": {
        $mergeObjects: ["$address", { "country": "USA" }]
      }
    }
  }
])
```

Ausgabe

```
[
  {
    "_id": 1,
    "combinedAddress": {
      "city": "New York",
      "state": "NY",
      "country": "USA"
    }
  },
  {
    "_id": 2,
    "combinedAddress": {
      "city": "Los Angeles",
      "state": "CA",
      "country": "USA"
    }
```

```
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$mergeObjects` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('collection');

  const result = await collection.aggregate([
    {
      $project: {
        "combinedAddress": {
          $mergeObjects: ["$address", { "country": "USA" }]
        }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
```

```
    db = client['test']
    collection = db['collection']

    result = list(collection.aggregate([
        {
            '$project': {
                "combinedAddress": {
                    "$mergeObjects": ["$address", { "country": "USA" }]
                }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$millisecond

Der `$millisecond` Operator in Amazon DocumentDB wird verwendet, um den Millisekundenanteil eines Datumswerts zu extrahieren.

Parameter

Keine

Beispiel (MongoDB Shell)

Dieses Beispiel zeigt, wie der `$millisecond` Operator verwendet wird, um den Millisekundenanteil eines Datumswerts zu extrahieren.

Erstellen Sie Beispieldokumente

```
db.events.insert([
  {
    "name": "Event 1",
    "timestamp": ISODate("2023-04-21T10:30:15.123Z")
  },
  {
    "name": "Event 2",
    "timestamp": ISODate("2023-04-21T10:30:15.456Z")
```

```
  },
  {
    "name": "Event 3",
    "timestamp": ISODate("2023-04-21T10:30:15.789Z")
  }
])
```

Beispiel für eine Abfrage

```
db.events.aggregate([
  {
    $project: {
      name: 1,
      milliseconds: { $millisecond: "$timestamp" }
    }
  }
])
```

Ausgabe

```
[
  {
    "_id": ObjectId("644332a42054ed1b0d15f0c1"),
    "name": "Event 1",
    "milliseconds": 123
  },
  {
    "_id": ObjectId("644332a42054ed1b0d15f0c2"),
    "name": "Event 2",
    "milliseconds": 456
  },
  {
    "_id": ObjectId("644332a42054ed1b0d15f0c3"),
    "name": "Event 3",
    "milliseconds": 789
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$millisecond` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const events = db.collection('events');

  const result = await events.aggregate([
    {
      $project: {
        name: 1,
        milliseconds: { $millisecond: '$timestamp' }
      }
    }
  ]).toArray();

  console.log(result);

  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
db = client['test']
events = db['events']

result = list(events.aggregate([
    {
        '$project': {
            'name': 1,
            'milliseconds': { '$millisecond': '$timestamp' }
        }
```

```
    }
]))

print(result)

client.close()
```

$min

Der `$min` Aggregationsoperator gibt den Mindestwert eines angegebenen Felds in einer Reihe von Dokumenten zurück. Dieser Operator ist nützlich, um den niedrigsten Wert in einer Reihe von Dokumenten zu finden.

Parameter

- `expression`: Der auszuwertende Ausdruck. Dies kann ein Feldpfad, eine Variable oder ein beliebiger Ausdruck sein, der in einen Wert aufgelöst wird.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung des `$min` Operators, um den Mindestwert des `age` Felds in mehreren Dokumenten zu ermitteln.

Erstellen Sie Beispieldokumente

```
db.users.insertMany([
  { name: "John", age: 35 },
  { name: "Jane", age: 28 },
  { name: "Bob", age: 42 },
  { name: "Alice", age: 31 }
]);
```

Beispiel für eine Abfrage

```
db.users.aggregate([
  { $group: { _id: null, minAge: { $min: "$age" } } },
  { $project: { _id: 0, minAge: 1 } }
])
```

Ausgabe

```
[ { minAge: 28 } ]
```

Beispiel für die Verwendung von Fensteroperatoren (MongoDB Shell)

Neu ab Version 8.0.2.

Der `$min` Operator kann auch als Fensterbediener auf der `$setWindowFields` Bühne verwendet werden. In diesem Zusammenhang gibt es den Mindestwert für die Dokumente in jedem Fenster zurück. Sie geben den Operator unter dem `output` Feld an und definieren optional die Fenstergrenzen mit einem `window` Dokument.

> **Note**
>
> Bei Verwendung als Fensteroperator in `$setWindowFields` `$min` ist er auf 100 MB an Zwischendaten begrenzt. Eine Operation, die dieses Limit überschreitet, gibt einen Fehler zurück.

Erstellen Sie Beispieldokumente

```
db.stockPrices.insertMany([
  { _id: 1, ticker: "ABC", hour: 1, price: 50 },
  { _id: 2, ticker: "ABC", hour: 2, price: 45 },
  { _id: 3, ticker: "ABC", hour: 3, price: 60 },
  { _id: 4, ticker: "ABC", hour: 4, price: 40 },
  { _id: 5, ticker: "XYZ", hour: 1, price: 30 },
  { _id: 6, ticker: "XYZ", hour: 2, price: 35 },
  { _id: 7, ticker: "XYZ", hour: 3, price: 25 }
]);
```

Beispiel für eine Abfrage

Das folgende Beispiel partitioniert die Dokumente nach`ticker`, sortiert jede Partition nach und gibt den niedrigsten Preis zurück`hour`, der vom Anfang der Partition bis zum aktuellen Dokument angezeigt wurde.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
```

```
      output: {
        lowestSoFar: {
          $min: "$price",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "ticker": "ABC", "hour": 1, "price": 50, "lowestSoFar": 50 },
  { "_id": 2, "ticker": "ABC", "hour": 2, "price": 45, "lowestSoFar": 45 },
  { "_id": 3, "ticker": "ABC", "hour": 3, "price": 60, "lowestSoFar": 45 },
  { "_id": 4, "ticker": "ABC", "hour": 4, "price": 40, "lowestSoFar": 40 },
  { "_id": 5, "ticker": "XYZ", "hour": 1, "price": 30, "lowestSoFar": 30 },
  { "_id": 6, "ticker": "XYZ", "hour": 2, "price": 35, "lowestSoFar": 30 },
  { "_id": 7, "ticker": "XYZ", "hour": 3, "price": 25, "lowestSoFar": 25 }
]
```

Jedes Dokument wird um das aktuelle Dokument erweitert`lowestSoFar`, wobei das aktuelle Minimum `price` innerhalb seiner Partition bis einschließlich des aktuellen Dokuments liegt.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$min` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten. Die folgenden Beispiele zeigen sowohl die Verwendung des Akkumulators (in`$group`) als auch die Verwendung von Fensteroperatoren (in`$setWindowFields`):

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
```

```
  // Accumulator usage: minimum value across a group
  const users = db.collection('users');
  await users.insertMany([
    { name: "John", age: 35 },
    { name: "Jane", age: 28 },
    { name: "Bob", age: 42 },
    { name: "Alice", age: 31 }
  ]);
  const accumulatorResult = await users.aggregate([
    { $group: { _id: null, minAge: { $min: "$age" } } }
  ]).toArray();
  console.log('Accumulator result:', accumulatorResult);

  // Window operator usage: running minimum within each partition
  const stockPrices = db.collection('stockPrices');
  await stockPrices.insertMany([
    { _id: 1, ticker: "ABC", hour: 1, price: 50 },
    { _id: 2, ticker: "ABC", hour: 2, price: 45 },
    { _id: 3, ticker: "ABC", hour: 3, price: 60 },
    { _id: 4, ticker: "ABC", hour: 4, price: 40 },
    { _id: 5, ticker: "XYZ", hour: 1, price: 30 },
    { _id: 6, ticker: "XYZ", hour: 2, price: 35 },
    { _id: 7, ticker: "XYZ", hour: 3, price: 25 }
  ]);
  const windowResult = await stockPrices.aggregate([
    {
      $setWindowFields: {
        partitionBy: "$ticker",
        sortBy: { hour: 1 },
        output: {
          lowestSoFar: {
            $min: "$price",
            window: { documents: ["unbounded", "current"] }
          }
        }
      }
    }
  ]).toArray();
  console.log('Window result:', windowResult);

  client.close();
}
```

```
example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']

    # Accumulator usage: minimum value across a group
    users = db.users
    users.insert_many([
        { "name": "John", "age": 35 },
        { "name": "Jane", "age": 28 },
        { "name": "Bob", "age": 42 },
        { "name": "Alice", "age": 31 }
    ])
    accumulator_result = list(users.aggregate([
        { "$group": { "_id": None, "minAge": { "$min": "$age" } } }
    ]))
    print('Accumulator result:', accumulator_result)

    # Window operator usage: running minimum within each partition
    stock_prices = db.stockPrices
    stock_prices.insert_many([
        { "_id": 1, "ticker": "ABC", "hour": 1, "price": 50 },
        { "_id": 2, "ticker": "ABC", "hour": 2, "price": 45 },
        { "_id": 3, "ticker": "ABC", "hour": 3, "price": 60 },
        { "_id": 4, "ticker": "ABC", "hour": 4, "price": 40 },
        { "_id": 5, "ticker": "XYZ", "hour": 1, "price": 30 },
        { "_id": 6, "ticker": "XYZ", "hour": 2, "price": 35 },
        { "_id": 7, "ticker": "XYZ", "hour": 3, "price": 25 }
    ])
    window_result = list(stock_prices.aggregate([
        {
            "$setWindowFields": {
                "partitionBy": "$ticker",
                "sortBy": { "hour": 1 },
                "output": {
                    "lowestSoFar": {
```

```
                        "$min": "$price",
                        "window": { "documents": ["unbounded", "current"] }
                    }
                }
            }
        }
    ]))
    print('Window result:', window_result)

    client.close()

example()
```

$ min

Neu ab Version 8.0.1.

Der `$minN` Operator in Amazon DocumentDB gibt die N kleinsten Werte zurück. Wenn er als Akkumulator in einer `$group` Stufe verwendet wird, gibt er ein Array mit den N Mindestwerten in jeder Gruppe zurück. Wenn er als Array-Ausdrucksoperator verwendet wird, gibt er die N minimalen Elemente eines Arrays zurück.

Parameter

- `input`: Ein Ausdruck, der in das Array oder Feld aufgelöst wird, aus dem die n Mindestwerte zurückgegeben werden sollen.
- n: Ein Ausdruck, der in eine positive Ganzzahl aufgelöst wird und angibt, wie viele Mindestwerte zurückgegeben werden sollen.

Behavior

Verhalten des Operators „Array-Ausdruck

- Sie können keinen Wert von n weniger als angeben`1`.
- `$minN`filtert die im `input` Array gefundenen `null` Werte heraus.
- Wenn der angegebene n Wert größer oder gleich der Anzahl der Elemente im `input` Array ist, werden alle Elemente im `input` Array `$minN` zurückgegeben.
- Wenn es in einen Wert `input` aufgelöst wird, der kein Array ist, tritt bei der Aggregationsoperation ein Fehler auf.

- Wenn es sowohl numerische als auch Zeichenkettenelemente `input` enthält, werden die numerischen Elemente gemäß der BSON-Vergleichsreihenfolge vor den Zeichenkettenelementen sortiert.

Verhalten des Akkumulators

- Bei Verwendung als Akkumulator n muss es sich um einen positiven Integralausdruck handeln, der entweder eine Konstante ist oder vom `_id` Wert für abhängt. `$group`
- `$minN`filtert Nullwerte und fehlende Werte heraus.
- Wenn die Gruppe weniger als n Elemente enthält, werden alle Elemente in der Gruppe `$minN` zurückgegeben.
- `$minN`vergleicht Eingabedaten gemäß der BSON-Vergleichsreihenfolge, um den geeigneten Ausgabetyp zu ermitteln. Wenn die Eingabedaten mehrere Datentypen enthalten, ist der `$minN` Ausgabetyp der niedrigste in der Vergleichsreihenfolge.

Reihenfolge der Ausgaben

`$minN`gibt Werte in keiner bestimmten Sortierreihenfolge zurück. Wenn die Garantie einer bestimmten Sortierreihenfolge erforderlich ist, verwenden Sie `$bottomN` stattdessen die Option oder schließen Sie das Ergebnis damit `$sortArray` ein.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$minN` Akkumulator verwendet wird, um die beiden niedrigsten Punktzahlen für jedes Subjekt abzurufen.

Erstellen Sie Beispieldokumente

```
db.scores.insertMany([
  { subject: "math", score: 85 },
  { subject: "math", score: 72 },
  { subject: "math", score: 93 },
  { subject: "math", score: 68 },
  { subject: "science", score: 90 },
  { subject: "science", score: 78 },
  { subject: "science", score: 65 },
  { subject: "science", score: 88 }
]);
```

Beispiel für eine Abfrage

```
db.scores.aggregate([
  { $group: { _id: "$subject", lowestTwo: { $minN: { input: "$score", n: 2 } } } }
]);
```

Ausgabe

```
[
  { "_id": "math", "lowestTwo": [72, 68] },
  { "_id": "science", "lowestTwo": [78, 65] }
]
```

Beispiel für die Verwendung von Ausdrücken (MongoDB Shell)

Der `$minN` Operator kann auch als Ausdruck innerhalb einer `$project` Stufe verwendet werden, um die N kleinsten Elemente aus einem Array-Feld zurückzugeben.

Erstellen Sie Beispieldokumente

```
db.readings.insertMany([
  { _id: 1, sensor: "A", values: [45, 12, 78, 3, 56] },
  { _id: 2, sensor: "B", values: [90, 23, 67, 11, 44] }
]);
```

Beispiel für eine Abfrage

```
db.readings.aggregate([
  { $project: {
      lowestThree: { $minN: { input: "$values", n: 3 } }
    }}
]);
```

Ausgabe

```
[
  { "_id": 1, "lowestThree": [45, 12, 3] },
  { "_id": 2, "lowestThree": [44, 23, 11] }
]
```

Beispiel für die Verwendung von Fensteroperatoren (MongoDB Shell)

Neu ab Version 8.0.2.

Der `$minN` Operator kann auch als Fensterbediener auf der `$setWindowFields` Bühne verwendet werden. In diesem Zusammenhang gibt er ein Array mit bis zu den n kleinsten Werten für die Dokumente in jedem Fenster zurück. Sie geben den Operator unter dem `output` Feld an und definieren optional die Fenstergrenzen mit einem `window` Dokument.

> **Note**
>
> Bei Verwendung als Fensteroperator in `$setWindowFields` `$minN` ist er auf 100 MB an Zwischendaten begrenzt. Eine Operation, die dieses Limit überschreitet, gibt einen Fehler zurück.

Erstellen Sie Beispieldokumente

```
db.stockPrices.insertMany([
  { _id: 1, ticker: "ABC", hour: 1, price: 50 },
  { _id: 2, ticker: "ABC", hour: 2, price: 45 },
  { _id: 3, ticker: "ABC", hour: 3, price: 60 },
  { _id: 4, ticker: "ABC", hour: 4, price: 40 },
  { _id: 5, ticker: "XYZ", hour: 1, price: 30 },
  { _id: 6, ticker: "XYZ", hour: 2, price: 35 },
  { _id: 7, ticker: "XYZ", hour: 3, price: 25 }
]);
```

Beispiel für eine Abfrage

Das folgende Beispiel partitioniert die Dokumente nach`ticker`, sortiert jede Partition nach und gibt die beiden niedrigsten Preise zurück`hour`, die vom Anfang der Partition bis zum aktuellen Dokument angezeigt wurden.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        lowestTwoSoFar: {
```

```
          $minN: { input: "$price", n: 2 },
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "ticker": "ABC", "hour": 1, "price": 50, "lowestTwoSoFar": [50] },
  { "_id": 2, "ticker": "ABC", "hour": 2, "price": 45, "lowestTwoSoFar": [45, 50] },
  { "_id": 3, "ticker": "ABC", "hour": 3, "price": 60, "lowestTwoSoFar": [45, 50] },
  { "_id": 4, "ticker": "ABC", "hour": 4, "price": 40, "lowestTwoSoFar": [40, 45] },
  { "_id": 5, "ticker": "XYZ", "hour": 1, "price": 30, "lowestTwoSoFar": [30] },
  { "_id": 6, "ticker": "XYZ", "hour": 2, "price": 35, "lowestTwoSoFar": [30, 35] },
  { "_id": 7, "ticker": "XYZ", "hour": 3, "price": 25, "lowestTwoSoFar": [25, 30] }
]
```

Jedes Dokument wird um die beiden kleinsten `price` Werte in seinem Fenster erweitert. `lowestTwoSoFar` Wie im Abschnitt Reihenfolge der Ausgaben erwähnt, ist die Reihenfolge der Werte innerhalb des zurückgegebenen Arrays nicht garantiert.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$minN` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten. Die folgenden Beispiele zeigen die Verwendung von Akkumulatoren (in`$group`), Ausdrücken (in`$project`) und Fensteroperatoren (in`$setWindowFields`):

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  try {
    await client.connect();
```

```
    const db = client.db('test');

    // Accumulator usage: N smallest values per group
    const scores = db.collection('scores');
    await scores.insertMany([
      { subject: "math", score: 85 },
      { subject: "math", score: 72 },
      { subject: "math", score: 93 },
      { subject: "math", score: 68 },
      { subject: "science", score: 90 },
      { subject: "science", score: 78 },
      { subject: "science", score: 65 },
      { subject: "science", score: 88 }
    ]);
    const accumulatorResult = await scores.aggregate([
      { $group: { _id: "$subject", lowestTwo: { $minN: { input: "$score", n:
2 } } } }
    ]).toArray();
    console.log('Accumulator result:', accumulatorResult);

    // Expression usage: N smallest elements from an array field
    const readings = db.collection('readings');
    await readings.insertMany([
      { _id: 1, sensor: "A", values: [45, 12, 78, 3, 56] },
      { _id: 2, sensor: "B", values: [90, 23, 67, 11, 44] }
    ]);
    const expressionResult = await readings.aggregate([
      { $project: { lowestThree: { $minN: { input: "$values", n: 3 } } } }
    ]).toArray();
    console.log('Expression result:', expressionResult);

    // Window operator usage: N smallest values over a window in each partition
    const stockPrices = db.collection('stockPrices');
    await stockPrices.insertMany([
      { _id: 1, ticker: "ABC", hour: 1, price: 50 },
      { _id: 2, ticker: "ABC", hour: 2, price: 45 },
      { _id: 3, ticker: "ABC", hour: 3, price: 60 },
      { _id: 4, ticker: "ABC", hour: 4, price: 40 },
      { _id: 5, ticker: "XYZ", hour: 1, price: 30 },
      { _id: 6, ticker: "XYZ", hour: 2, price: 35 },
      { _id: 7, ticker: "XYZ", hour: 3, price: 25 }
    ]);
    const windowResult = await stockPrices.aggregate([
      {
```

```
        $setWindowFields: {
          partitionBy: "$ticker",
          sortBy: { hour: 1 },
          output: {
            lowestTwoSoFar: {
              $minN: { input: "$price", n: 2 },
              window: { documents: ["unbounded", "current"] }
            }
          }
        }
      }
    ]).toArray();
    console.log('Window result:', windowResult);

  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    try:
        db = client['test']

        # Accumulator usage: N smallest values per group
        scores = db['scores']
        scores.insert_many([
            { 'subject': 'math', 'score': 85 },
            { 'subject': 'math', 'score': 72 },
            { 'subject': 'math', 'score': 93 },
            { 'subject': 'math', 'score': 68 },
            { 'subject': 'science', 'score': 90 },
            { 'subject': 'science', 'score': 78 },
            { 'subject': 'science', 'score': 65 },
```

```
            { 'subject': 'science', 'score': 88 }
        ])
        accumulator_result = list(scores.aggregate([
            { '$group': { '_id': '$subject', 'lowestTwo': { '$minN': { 'input':
'$score', 'n': 2 } } } }
        ]))
        print('Accumulator result:', accumulator_result)

        # Expression usage: N smallest elements from an array field
        readings = db['readings']
        readings.insert_many([
            { '_id': 1, 'sensor': 'A', 'values': [45, 12, 78, 3, 56] },
            { '_id': 2, 'sensor': 'B', 'values': [90, 23, 67, 11, 44] }
        ])
        expression_result = list(readings.aggregate([
            { '$project': { 'lowestThree': { '$minN': { 'input': '$values', 'n':
3 } } } }
        ]))
        print('Expression result:', expression_result)

        # Window operator usage: N smallest values over a window in each partition
        stock_prices = db['stockPrices']
        stock_prices.insert_many([
            { '_id': 1, 'ticker': 'ABC', 'hour': 1, 'price': 50 },
            { '_id': 2, 'ticker': 'ABC', 'hour': 2, 'price': 45 },
            { '_id': 3, 'ticker': 'ABC', 'hour': 3, 'price': 60 },
            { '_id': 4, 'ticker': 'ABC', 'hour': 4, 'price': 40 },
            { '_id': 5, 'ticker': 'XYZ', 'hour': 1, 'price': 30 },
            { '_id': 6, 'ticker': 'XYZ', 'hour': 2, 'price': 35 },
            { '_id': 7, 'ticker': 'XYZ', 'hour': 3, 'price': 25 }
        ])
        window_result = list(stock_prices.aggregate([
            {
                '$setWindowFields': {
                    'partitionBy': '$ticker',
                    'sortBy': { 'hour': 1 },
                    'output': {
                        'lowestTwoSoFar': {
                            '$minN': { 'input': '$price', 'n': 2 },
                            'window': { 'documents': ['unbounded', 'current'] }
                        }
                    }
                }
            }
```

```
        ]))
        print('Window result:', window_result)

    finally:
        client.close()

example()
```

$minute

Die `$minute` Aggregationspipeline-Phase in Amazon DocumentDB extrahiert den Minutenwert aus einem Datums- oder Zeitstempelfeld.

Dieser Operator ist nützlich, wenn Sie datums- und zeitbasierte Berechnungen oder Gruppierungen innerhalb Ihrer Aggregationspipeline durchführen müssen.

Parameter

- `expression`: Das Datums- oder Zeitstempelfeld, aus dem der Minutenwert extrahiert werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$minute` Operator verwendet wird, um die Dokumente nach dem aus dem Zeitstempelfeld extrahierten Minutenwert zu gruppieren und die Anzahl der Dokumente in jeder Gruppe zu zählen.

Erstellen Sie Beispieldokumente

```
db.events.insertMany([
  { timestamp: new Date("2023-04-15T10:30:25.000Z") },
  { timestamp: new Date("2023-04-15T10:30:35.000Z") },
  { timestamp: new Date("2023-04-15T10:31:05.000Z") },
  { timestamp: new Date("2023-04-15T10:31:45.000Z") },
  { timestamp: new Date("2023-04-15T10:32:15.000Z") }
]);
```

Beispiel für eine Abfrage

```
db.events.aggregate([
  {
```

```
    $group: {
      _id: {
        minute: { $minute: "$timestamp" }
      },
      count: { $count: {} }
    }
  },
  { $sort: { "_id.minute": 1 } }
]);
```

Ausgabe

```
[
  { "_id": { "minute": 30 }, "count": 2 },
  { "_id": { "minute": 31 }, "count": 2 },
  { "_id": { "minute": 32 }, "count": 1 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$minute` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('events');

  await collection.insertMany([
    { timestamp: new Date("2023-04-15T10:30:25.000Z") },
    { timestamp: new Date("2023-04-15T10:30:35.000Z") },
    { timestamp: new Date("2023-04-15T10:31:05.000Z") },
    { timestamp: new Date("2023-04-15T10:31:45.000Z") },
    { timestamp: new Date("2023-04-15T10:32:15.000Z") }
  ]);

  const result = await collection.aggregate([
```

```
    {
      $group: {
        _id: {
          minute: { $minute: "$timestamp" }
        },
        count: { $count: {} }
      }
    },
    { $sort: { "_id.minute": 1 } }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient
from datetime import datetime

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['events']

    collection.insert_many([
        {'timestamp': datetime(2023, 4, 15, 10, 30, 25)},
        {'timestamp': datetime(2023, 4, 15, 10, 30, 35)},
        {'timestamp': datetime(2023, 4, 15, 10, 31, 5)},
        {'timestamp': datetime(2023, 4, 15, 10, 31, 45)},
        {'timestamp': datetime(2023, 4, 15, 10, 32, 15)}
    ])

    result = list(collection.aggregate([
        {
            '$group': {
                '_id': {
                    'minute': {'$minute': '$timestamp'}
                },
```

```
                'count': {'$count': {}}
            }
        },
        {'$sort': {'_id.minute': 1}}
    ]))

    print(result)
    client.close()

example()
```

$mod

Der $mod Operator ist ein arithmetischer Operator, der eine Modulo-Operation für eine Zahl ausführt. Er gibt den Rest der Division einer Zahl durch eine andere zurück. Dieser Operator wird häufig verwendet, um zu bestimmen, ob eine Zahl ungerade oder gerade ist, oder um Elemente in eine endliche Anzahl von Gruppen zu verteilen.

Parameter

- `expression1`: Der Dividendenausdruck.
- `expression2`: Der Divisor-Ausdruck.

Beispiel (MongoDB Shell)

Dieses Beispiel zeigt, wie der $mod Operator verwendet wird, um die Anzahl der übrig gebliebenen Widgets zu ermitteln, wenn Pakete mit 100 Stück versendet werden.

Erstellen Sie Beispieldokumente

```
db.widgets.insertMany([
  { "_id" : 1, "widget" : "A", "count" : 80372 },
  { "_id" : 2, "widget" : "B", "count" : 409282 },
  { "_id" : 3, "widget" : "C", "count" : 60739 }
])
```

Beispiel für eine Abfrage

```
db.widgets.aggregate([
  { $addFields: { leftOver: { $mod: [ "$count", 100 ] } } }
```

```
])
```

Ausgabe

```
[
  { "_id" : 1, "widget" : "A", "count" : 80372, "leftOver" : 72 },
  { "_id" : 2, "widget" : "B", "count" : 409282, "leftOver" : 82 },
  { "_id" : 3, "widget" : "C", "count" : 60739, "leftOver" : 39 }
]
```

Die Ausgabe zeigt den Rest `count` geteilt durch 100 für jedes Dokument. Dies entspricht der Anzahl der übrig gebliebenen Widgets beim Versand in Paketen mit 100 Stück.

Codebeispiele

Um ein Codebeispiel für die Verwendung des $mod Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const widgets = db.collection('widgets');

  await widgets.insertMany([
    { "_id" : 1, "widget" : "A", "count" : 80372 },
    { "_id" : 2, "widget" : "B", "count" : 409282 },
    { "_id" : 3, "widget" : "C", "count" : 60739 }
  ]);

  const result = await widgets.aggregate([
    { $addFields: { leftOver: { $mod: [ "$count", 100 ] } } }
  ]).toArray();

  console.log(result);
  client.close();
}
```

```
main();
```

Python

```
from pymongo import MongoClient

client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
db = client['test']
widgets = db['widgets']

widgets.insert_many([
    { "_id" : 1, "widget" : "A", "count" : 80372 },
    { "_id" : 2, "widget" : "B", "count" : 409282 },
    { "_id" : 3, "widget" : "C", "count" : 60739 }
])

result = list(widgets.aggregate([
    { "$addFields": { "leftOver": { "$mod": [ "$count", 100 ] } } }
]))

print(result)
client.close()
```

$month

Der `$month` Operator in Amazon DocumentDB gibt den Monat eines Datums als Zahl zwischen 1 und 12 zurück. Dieser Operator ist nützlich, um die Monatskomponente aus einem Datumsfeld zu extrahieren und datumsbasierte Aggregationen und Analysen durchzuführen.

Parameter

- `date_expression`: Dies ist der Ausdruck oder das Feld, das das Datum oder den Zeitstempel enthält, aus dem Sie den Monat extrahieren möchten.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$month` Operator verwendet wird, um den Monat aus einem Datumsfeld zu extrahieren und die Daten nach Monat zu gruppieren.

Erstellen Sie Beispieldokumente

```
db.sales.insert([
  { product: "abc123", price: 10.99, date: new Date("2022-01-15") },
  { product: "def456", price: 15.50, date: new Date("2022-02-28") },
  { product: "ghi789", price: 8.25, date: new Date("2022-03-10") },
  { product: "jkl012", price: 12.75, date: new Date("2022-04-05") },
  { product: "mno345", price: 18.99, date: new Date("2022-05-20") }
]);
```

Beispiel für eine Abfrage

```
db.sales.aggregate([
  { $group: {
      _id: { month: { $month: "$date" } },
      totalSales: { $sum: "$price" }
    }},
  { $sort: { "_id.month": 1 } }
]);
```

Ausgabe

```
[
  { _id: { month: 1 }, totalSales: 10.99 },
  { _id: { month: 2 }, totalSales: 15.5 },
  { _id: { month: 3 }, totalSales: 8.25 },
  { _id: { month: 4 }, totalSales: 12.75 },
  { _id: { month: 5 }, totalSales: 18.99 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$month` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function groupSalesByMonth() {
```

```
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  try {
    await client.connect();
    const db = client.db('test');
    const collection = db.collection('sales');

    const pipeline = [
      {
        $group: {
          _id: { month: { $month: "$date" } },
          totalSales: { $sum: "$price" }
        }
      },
      {
        $sort: { "_id.month": 1 }
      }
    ];

    const results = await collection.aggregate(pipeline).toArray();

    console.dir(results, { depth: null });

  } finally {
    await client.close();
  }
}

groupSalesByMonth().catch(console.error);
```

Python

```
from pymongo import MongoClient

def group_sales_by_month():

    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    try:
```

```
        db = client.test
        collection = db.sales

        pipeline = [
            {
                "$group": {
                    "_id": { "$month": "$date" },
                    "totalSales": { "$sum": "$price" }
                }
            },
            {
                "$sort": { "_id": 1 }
            }
        ]

        results = collection.aggregate(pipeline)

        for doc in results:
            print(doc)

    except Exception as e:
        print(f"An error occurred: {e}")

    finally:
        client.close()

group_sales_by_month()
```

$multiply

Der `$multiply` Operator in Amazon DocumentDB wird verwendet, um die Werte von zwei oder mehr Feldern oder Ausdrücken zu multiplizieren. Dieser Operator ist besonders nützlich, um arithmetische Operationen an numerischen Feldern in Dokumenten durchzuführen. Er kann in verschiedenen Phasen der Aggregationspipeline eingesetzt werden, z. B. bei `$project` und`$addFields`, um neue Felder zu erstellen oder bestehende zu ändern.

Parameter

- `expression1`: Der erste numerische Ausdruck, der multipliziert werden soll.
- `expression2`: Der zweite numerische Ausdruck, der multipliziert werden soll.

- `[expression3, ...]`: (optional) Zusätzliche numerische Ausdrücke, die multipliziert werden sollen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung `$multiply` zur Berechnung `bonus_miles` durch Multiplikation `base_miles` und `bonus_rate` für Kunden, die für die Reise eine Kreditkarte verwendet haben.

Beispieldokumente

```
db.miles.insertMany([
{ "_id": 1, "customer_name": "Arnav Desai", "member_since":
 ISODate("1997-03-01T00:00:00Z"), "base_miles": 2500, "bonus_rate": 1.8, "credit_card":
 true, "trip_cost": 250 },
{ "_id": 2, "customer_name": "Jorge Souza", "member_since":
 ISODate("2004-01-10T00:00:00Z"), "base_miles": 1890, "bonus_rate": 1.4, "credit_card":
 true, "trip_cost": 189 },
{ "_id": 3, "customer_name": "Saanvi Sarkar", "member_since":
 ISODate("1999-11-22T00:00:00Z"), "base_miles": 3250, "bonus_rate": 1.8, "credit_card":
 false, "trip_cost": 325 },
{ "_id": 4, "customer_name": "Paulo Santos", "member_since":
 ISODate("2021-06-19T00:00:00Z"), "base_miles": 2980, "bonus_rate": 1.2, "credit_card":
 true, "trip_cost": 298 },
{ "_id": 5, "customer_name": "Wang Xiulan", "member_since":
 ISODate("1995-12-04T00:00:00Z"), "base_miles": 1350, "bonus_rate": 1.9, "credit_card":
 false, "trip_cost": 135 }
]);
```

Beispiel für eine Abfrage

```
db.miles.aggregate([
  {
    $match: { credit_card: true }
  },
  {
    $project: {
      customer_name: 1,
      base_miles: 1,
      bonus_rate:1,
      credit_card: 1,
      total_miles: {
```

```
        $multiply: ["$base_miles", "$bonus_rate"]
      }
    }
  }
]);
```

Ausgabe

```
[
  { _id: 1, customer_name: 'Arnav Desai', base_miles: 12500, bonus_rate: 1.8,
 credit_card: true, total_miles: 22500 },
  { _id: 3, customer_name: 'Saanvi Sarkar',base_miles: 15200, bonus_rate: 1.8,
 credit_card: true, total_miles: 27360 },
  { _id: 4, customer_name: 'Paulo Santos', base_miles: 3400, bonus_rate: 1.1,
 credit_card: true, total_miles: 3740 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$multiply` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function multiplyBonusMiles() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('miles');

  const result = await collection.aggregate([
    { $match: { credit_card: true } },
    {
      $project: {
        customer_name: 1,
        base_miles: 1,
        bonus_rate: 1,
        credit_card: 1,
        total_miles: {
```

```
          $multiply: ["$base_miles", "$bonus_rate"]
        }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

multiplyBonusMiles();
```

Python

```
from pymongo import MongoClient

def multiply_bonus_miles():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['miles']

    result = list(collection.aggregate([
        {'$match': {'credit_card': True}},
        {
            '$project': {
                'customer_name': 1,
                'base_miles': 1,
                'bonus_rate': 1,
                'credit_card': 1,
                'total_miles': {
                    '$multiply': ['$base_miles', '$bonus_rate']
                }
            }
        }
    ]))

    print(result)
    client.close()

multiply_bonus_miles()
```

$ natürlich

Der `$natural` Operator in Amazon DocumentDB wird verwendet, um die Dokumente in ihrer natürlichen Reihenfolge zu sortieren, d. h. der Reihenfolge, in der sie in die Sammlung eingefügt wurden. Dies steht im Gegensatz zum Standard-Sortierverhalten, bei dem die Dokumente anhand der Werte der angegebenen Felder sortiert werden.

Parameter

Keine

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$natural` Operator verwendet wird, um die Dokumente in einer Sammlung in ihrer natürlichen Reihenfolge zu sortieren.

Erstellen Sie Beispieldokumente

```
db.people.insertMany([
  { "_id": 1, "name": "María García", "age": 28 },
  { "_id": 2, "name": "Arnav Desai", "age": 32 },
  { "_id": 3, "name": "Li Juan", "age": 25 },
  { "_id": 4, "name": "Carlos Salazar", "age": 41 },
  { "_id": 5, "name": "Sofia Martínez", "age": 35 }
]);
```

Beispiel für eine Abfrage

```
db.people.find({}, { "_id": 1, "name": 1 }).sort({ "$natural": 1 });
```

Ausgabe

```
[
  { "_id": 1, "name": "María García" },
  { "_id": 2, "name": "Arnav Desai" },
  { "_id": 3, "name": "Li Juan" },
  { "_id": 4, "name": "Carlos Salazar" },
  { "_id": 5, "name": "Sofia Martínez" }
]
```

Die Abfrage sortiert die Dokumente in der Sammlung in ihrer natürlichen Reihenfolge, d. h. der Reihenfolge, in der sie eingefügt wurden.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$natural` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('people');

  const documents = await collection.find({}, { projection: { _id: 1, name: 1 } })
    .sort({ $natural: 1 })
    .toArray();

  console.log(documents);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['people']

    documents = list(collection.find({}, {'_id': 1, 'name': 1}).sort('$natural', 1))
    print(documents)

    client.close()
```

```
example()
```

$ne

Der `$ne` Aggregationsoperator vergleicht zwei Werte und gibt zurück, `true` wenn sie nicht gleich sind, andernfalls gibt er zurück`false`.

Parameter

- `expression1`: Der erste Wert, der verglichen werden soll.
- `expression2`: Der zweite zu vergleichende Wert.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$ne` Operator verwendet wird, um Bestellungen mit Statusänderungen zu identifizieren.

Erstellen Sie Beispieldokumente

```
db.orders.insertMany([
  { _id: 1, orderId: "A123", status: "shipped", expectedStatus: "shipped" },
  { _id: 2, orderId: "B456", status: "pending", expectedStatus: "shipped" },
  { _id: 3, orderId: "C789", status: "delivered", expectedStatus: "delivered" }
]);
```

Beispiel für eine Abfrage

```
db.orders.aggregate([
  {
    $project: {
      orderId: 1,
      status: 1,
      expectedStatus: 1,
      needsAttention: { $ne: ["$status", "$expectedStatus"] }
    }
  }
]);
```

Ausgabe

```
[
  { _id: 1, orderId: 'A123', status: 'shipped', expectedStatus: 'shipped',
 needsAttention: false },
  { _id: 2, orderId: 'B456', status: 'pending', expectedStatus: 'shipped',
 needsAttention: true },
  { _id: 3, orderId: 'C789', status: 'delivered', expectedStatus: 'delivered',
 needsAttention: false }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des $ne Aggregationsoperators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('orders');

  const result = await collection.aggregate([
    {
      $project: {
        orderId: 1,
        status: 1,
        expectedStatus: 1,
        needsAttention: { $ne: ["$status", "$expectedStatus"] }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['orders']

    result = list(collection.aggregate([
        {
            '$project': {
                'orderId': 1,
                'status': 1,
                'expectedStatus': 1,
                'needsAttention': { '$ne': ['$status', '$expectedStatus'] }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$not

Der `$not` Aggregationsoperator führt eine logische NOT-Operation für einen Ausdruck aus. Er gibt zurück`true`, ob der Ausdruck zu `false` ausgewertet wird und `false` ob der Ausdruck zu ausgewertet wird. `true`

Parameter

- `expression`: Der Ausdruck, der negiert werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$not` Operator verwendet wird, um boolesche Werte umzukehren.

Erstellen Sie Beispieldokumente

```
db.users.insertMany([
  { _id: 1, name: "Alice", active: true },
  { _id: 2, name: "Bob", active: false },
  { _id: 3, name: "Charlie", active: true }
]);
```

Beispiel für eine Abfrage

```
db.users.aggregate([
  {
    $project: {
      name: 1,
      active: 1,
      inactive: { $not: ["$active"] }
    }
  }
]);
```

Ausgabe

```
[
  { _id: 1, name: 'Alice', active: true, inactive: false },
  { _id: 2, name: 'Bob', active: false, inactive: true },
  { _id: 3, name: 'Charlie', active: true, inactive: false }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$not` Aggregationsoperators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
```

```
  const collection = db.collection('users');

  const result = await collection.aggregate([
    {
      $project: {
        name: 1,
        active: 1,
        inactive: { $not: ["$active"] }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['users']

    result = list(collection.aggregate([
        {
            '$project': {
                'name': 1,
                'active': 1,
                'inactive': { '$not': ['$active'] }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$objekt ToArray

Der `$objectToArray` Aggregationsoperator in Amazon DocumentDB konvertiert ein Objekt (oder Dokument) in ein Array. Die Eingabe für den Operator ist ein Dokument, und die Ausgabe besteht aus einem Array-Element für jedes Feld-Wert-Paar im Eingabedokument. Dieser Operator ist nützlich, wenn Sie mit den einzelnen Feldern eines Dokuments als Array arbeiten müssen, z. B. wenn Sie das Dokument mit dem Höchst- oder Minimalwert für ein bestimmtes Feld suchen möchten.

Parameter

- `expression`: Der Dokumentausdruck, der in ein Array konvertiert werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$objectToArray` Operator verwendet werden kann, um das Dokument mit dem maximalen Inventar für eine Videoverleihkette zu finden.

Erstellen Sie Beispieldokumente

```
db.videos.insertMany([
  {
    "_id": 1,
    "name": "Live Soft",
    "inventory": {
      "Des Moines": 1000,
      "Ames": 500
    }
  },
  {
    "_id": 2,
    "name": "Top Pilot",
    "inventory": {
      "Mason City": 250,
      "Des Moines": 1000
    }
  },
  {
    "_id": 3,
    "name": "Romancing the Rock",
    "inventory": {
      "Mason City": 250,
      "Ames": 500
```

```
    }
  },
  {
    "_id": 4,
    "name": "Bravemind",
    "inventory": {
      "Mason City": 250,
      "Des Moines": 1000,
      "Ames": 500
    }
  }
]);
```

Beispiel für eine Abfrage

```
db.videos.aggregate([
  {
    $project: {
      name: 1,
      videos: {
        $objectToArray: "$inventory"
      }
    }
  },
  {
    $unwind: "$videos"
  },
  {
    $group: {
      _id: "$name",
      maxInventory: {
        $max: "$videos.v"
      }
    }
  }
]);
```

Ausgabe

```
[
  {
    "_id": "Bravemind",
    "maxInventory": 1000
```

```
  },
  {
    "_id": "Live Soft",
    "maxInventory": 1000
  },
  {
    "_id": "Romancing the Rock",
    "maxInventory": 500
  },
  {
    "_id": "Top Pilot",
    "maxInventory": 1000
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$objectToArray` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function findMaxInventory() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const videos = db.collection('videos');

  const result = await videos.aggregate([
    {
      $project: {
        name: 1,
        videos: {
          $objectToArray: "$inventory"
        }
      }
    },
    {
      $unwind: "$videos"
    },
```

```
    {
      $group: {
        _id: "$name",
        maxInventory: {
          $max: "$videos.v"
        }
      }
    }
  ]).toArray();

  console.log(result);
  client.close();
}

findMaxInventory();
```

Python

```
from pymongo import MongoClient

def find_max_inventory():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    videos = db['videos']

    result = list(videos.aggregate([
        {
            '$project': {
                'name': 1,
                'videos': {
                    '$objectToArray': '$inventory'
                }
            }
        },
        {
            '$unwind': '$videos'
        },
        {
            '$group': {
                '_id': '$name',
                'maxInventory': {
```

```
                    '$max': '$videos.v'
                }
            }
        }
    ]))

    print(result)
    client.close()

find_max_inventory()
```

$or

Der `$or` Aggregationsoperator wertet mehrere Ausdrücke aus und gibt zurück, `true` ob mindestens ein Ausdruck zu ausgewertet wird. `true` Er gibt `false` nur zurück, wenn es bei allen Ausdrücken der Fall ist. `false`

Parameter

- `expressions`: Eine Reihe von Ausdrücken, die ausgewertet werden sollen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$or` Operator verwendet wird, um zu überprüfen, ob Produkte eines von mehreren Kriterien erfüllen.

Erstellen Sie Beispieldokumente

```
db.items.insertMany([
  { _id: 1, name: "Widget", price: 150, onSale: false },
  { _id: 2, name: "Gadget", price: 45, onSale: false },
  { _id: 3, name: "Tool", price: 200, onSale: true }
]);
```

Beispiel für eine Abfrage

```
db.items.aggregate([
  {
    $project: {
      name: 1,
```

```
      price: 1,
      onSale: 1,
      goodDeal: {
        $or: [
          { $lt: ["$price", 50] },
          { $eq: ["$onSale", true] }
        ]
      }
    }
  }
]);
```

Ausgabe

```
[
  { _id: 1, name: 'Widget', price: 150, onSale: false, goodDeal: false },
  { _id: 2, name: 'Gadget', price: 45, onSale: false, goodDeal: true },
  { _id: 3, name: 'Tool', price: 200, onSale: true, goodDeal: true }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$or` Aggregationsoperators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('items');

  const result = await collection.aggregate([
    {
      $project: {
        name: 1,
        price: 1,
        onSale: 1,
        goodDeal: {
```

```
          $or: [
            { $lt: ["$price", 50] },
            { $eq: ["$onSale", true] }
          ]
        }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['items']

    result = list(collection.aggregate([
        {
            '$project': {
                'name': 1,
                'price': 1,
                'onSale': 1,
                'goodDeal': {
                    '$or': [
                        { '$lt': ['$price', 50] },
                        { '$eq': ['$onSale', True] }
                    ]
                }
            }
        }
    ]))

    print(result)
```

```
    client.close()

example()
```

$out

Der `$out` Operator in Amazon DocumentDB wird verwendet, um das Ergebnis einer Aggregationspipeline in eine angegebene Sammlung zu schreiben.

`$out`sollte die letzte Phase in der Pipeline sein.

Parameter

- `output_collection`: Der Name der Ausgabesammlung, in die die Aggregationsergebnisse geschrieben werden sollen.

Hinweis: Wenn die Sammlung bereits existiert, wird sie durch die Ergebnisse der Aggregationsphase ersetzt.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$out` Operator in Amazon DocumentDB verwendet wird, um die Ergebnisse einer Aggregationspipeline in eine neue Sammlung zu schreiben.

Erstellen Sie Beispieldokumente

```
db.products.insertMany([
  { _id: 1, name: "Wireless Headphones", category: "Electronics", price: 100.0 },
  { _id: 2, name: "Smartphone", category: "Electronics", price: 200.0 },
  { _id: 3, name: "JavaScript Guide", category: "Books", price: 50.0 },
  { _id: 4, name: "Database Design Handbook", category: "Books", price: 75.0 }
]);
```

Beispiel für eine Abfrage

```
db.products.aggregate([
  { $group: { _id: "$category", totalPrice: { $sum: "$price" } } },
  { $out: "product_categories" }
])
```

Ausgabe

Keine (die Ergebnisse werden in die Ausgabesammlung geschrieben).

Die Aggregationspipeline gruppiert die Produkte nach Kategorien und berechnet den Gesamtpreis der Artikel für jede Kategorie. Der `$out` Operator schreibt die Ergebnisse in eine neue Sammlung mit dem Namen „product_categories“.

So zeigen Sie die Ergebnisse in der Ausgabesammlung an:

```
db.product_categories.find()
[
{ "_id" : "Books", "totalPrice" : 125 },
{ "_id" : "Electronics", "totalPrice" : 300 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$out` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function demo_out_operator() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const products = db.collection('products');

  // Execute aggregation with $out - results are stored in 'product_categories'
 collection
  await products.aggregate([
    { $group: { _id: "$category", totalPrice: { $sum: "$price" } } },
    { $out: "product_categories" }
  ]).toArray();

  // Retrieve the results from the output collection (limited to 20 records)
  const productCategories = db.collection('product_categories');
  const results = await productCategories.find({}).limit(20).toArray();
```

```
  console.log('Results stored in product_categories collection:', results);
  await client.close();
}

demo_out_operator();
```

Python

```
from pymongo import MongoClient

def demo_out_operator():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    products = db['products']

    # Execute aggregation with $out - results are stored in 'product_categories'
 collection
    list(products.aggregate([
        { '$group': { '_id': '$category', 'totalPrice': { '$sum': '$price' } } },
        { '$out': 'product_categories' }
    ]))

    # Retrieve the results from the output collection (limited to 20 records)
    product_categories = db['product_categories']
    results = list(product_categories.find({}).limit(20))

    print('Results stored in product_categories collection:', results)
    client.close()

demo_out_operator()
```

$ Perzentil

Neu ab Version 8.0.1.

Der `$percentile` Operator in Amazon DocumentDB berechnet angegebene Perzentilwerte für numerische Daten. Als Akkumulator berechnet er in der Phase einer Aggregationspipeline Perzentilwerte für Dokumente innerhalb einer Gruppe. `$group` Als Ausdruck berechnet er Perzentile für eine Reihe von Zahlen.

Parameter

- `input`: Ein Ausdruck, der in einen numerischen Wert oder ein Array numerischer Werte aufgelöst wird.
- `p`: Ein Array von Perzentilwerten zwischen 0 und 1, wobei jeder Wert ein zu berechnendes Perzentil darstellt. `[0.25, 0.5, 0.75]`Berechnet beispielsweise das 25., 50. und 75. Perzentil.
- `method`: Eine Zeichenfolge, die die Berechnungsmethode angibt. Derzeit wird nur `"approximate"` unterstützt.

Behavior

Die `"approximate"` Methode verwendet den T-Digest-Algorithmus zur Berechnung ungefährer, perzentilbasierter Metriken. Bei kleinen Datensätzen können unterschiedliche Perzentilwerte zu demselben Ergebnis führen. Beispiel: Bei nur 5 Werten pro Gruppe geben p90 und p99 möglicherweise beide den Maximalwert in der Gruppe zurück. Die Genauigkeit verbessert sich, wenn die Anzahl der Datenpunkte zunimmt.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$percentile` Operator verwendet wird, um das 25. und 75. Perzentil der Ergebnisse pro Klasse zu berechnen.

Erstellen Sie Beispieldokumente

```
db.students.insertMany([
  { class: "A", score: 72 },
  { class: "A", score: 85 },
  { class: "A", score: 90 },
  { class: "A", score: 68 },
  { class: "A", score: 95 },
  { class: "B", score: 80 },
  { class: "B", score: 75 },
  { class: "B", score: 92 },
  { class: "B", score: 88 },
  { class: "B", score: 70 }
]);
```

Beispiel für eine Abfrage

```
db.students.aggregate([
```

```
  { $group: {
      _id: "$class",
      percentiles: { $percentile: { input: "$score", p: [0.25, 0.75], method:
 "approximate" } }
    }}
]);
```

Ausgabe

```
[
  { "_id": "A", "percentiles": [72, 90] },
  { "_id": "B", "percentiles": [75, 88] }
]
```

Beispiel für die Verwendung von Ausdrücken (MongoDB Shell)

Der `$percentile` Operator kann auch als Ausdruck innerhalb einer `$project` Stufe verwendet werden, um Perzentile eines Array-Feldes zu berechnen.

Erstellen Sie Beispieldokumente

```
db.surveys.insertMany([
  { _id: 1, responses: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20] },
  { _id: 2, responses: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19] }
]);
```

Beispiel für eine Abfrage

```
db.surveys.aggregate([
  { $project: {
      quartiles: { $percentile: { input: "$responses", p: [0.25, 0.5, 0.75], method:
 "approximate" } }
    }}
]);
```

Ausgabe

```
[
  { "_id": 1, "quartiles": [6, 10, 16] },
  { "_id": 2, "quartiles": [5, 9, 15] }
]
```

Beispiel für die Verwendung von Fensteroperatoren (MongoDB Shell)

Neu ab Version 8.0.2.

Der `$percentile` Operator kann auch als Fensterbediener auf der `$setWindowFields` Bühne verwendet werden. In diesem Zusammenhang berechnet er die angegebenen Perzentilwerte für die Dokumente in jedem Fenster. Sie geben den Operator unter dem `output` Feld an und definieren optional die Fenstergrenzen mit einem Dokument. `window`

> **Note**
>
> Bei Verwendung als Fensteroperator in `$setWindowFields` `$percentile` ist er auf 100 MB an Zwischendaten begrenzt. Eine Operation, die dieses Limit überschreitet, gibt einen Fehler zurück.

Erstellen Sie Beispieldokumente

```
db.sensorData.insertMany([
  { _id: 1, sensor: "A", reading: 72 },
  { _id: 2, sensor: "A", reading: 85 },
  { _id: 3, sensor: "A", reading: 90 },
  { _id: 4, sensor: "A", reading: 68 },
  { _id: 5, sensor: "A", reading: 95 },
  { _id: 6, sensor: "B", reading: 80 },
  { _id: 7, sensor: "B", reading: 75 },
  { _id: 8, sensor: "B", reading: 92 },
  { _id: 9, sensor: "B", reading: 88 },
  { _id: 10, sensor: "B", reading: 70 }
]);
```

Beispiel für eine Abfrage

Im folgenden Beispiel werden die Dokumente nach `sensor` `reading` allen Dokumenten in jeder Partition partitioniert und das 25. und 75. Perzentil berechnet.

```
db.sensorData.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$sensor",
      output: {
        readingPercentiles: {
```

```
          $percentile: { input: "$reading", p: [0.25, 0.75], method: "approximate" },
          window: { documents: ["unbounded", "unbounded"] }
        }
      }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "sensor": "A", "reading": 72, "readingPercentiles": [72, 90] },
  { "_id": 2, "sensor": "A", "reading": 85, "readingPercentiles": [72, 90] },
  { "_id": 3, "sensor": "A", "reading": 90, "readingPercentiles": [72, 90] },
  { "_id": 4, "sensor": "A", "reading": 68, "readingPercentiles": [72, 90] },
  { "_id": 5, "sensor": "A", "reading": 95, "readingPercentiles": [72, 90] },
  { "_id": 6, "sensor": "B", "reading": 80, "readingPercentiles": [75, 88] },
  { "_id": 7, "sensor": "B", "reading": 75, "readingPercentiles": [75, 88] },
  { "_id": 8, "sensor": "B", "reading": 92, "readingPercentiles": [75, 88] },
  { "_id": 9, "sensor": "B", "reading": 88, "readingPercentiles": [75, 88] },
  { "_id": 10, "sensor": "B", "reading": 70, "readingPercentiles": [75, 88] }
]
```

Jedes Dokument wird um das ungefähre 25. und 75. Perzentil aller Dokumente in seiner Partition erweitert. `readingPercentiles` `reading`

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$percentile` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten. Die folgenden Beispiele zeigen die Verwendung von Akkumulatoren (in`$group`), Ausdrücken (in`$project`) und Fensteroperatoren (in`$setWindowFields`):

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const uri = 'mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false';
  const client = new MongoClient(uri);
```

```
 try {
   await client.connect();
   const db = client.db('test');

   // Accumulator usage: percentiles across grouped documents
   const students = db.collection('students');
   await students.insertMany([
     { class: "A", score: 72 },
     { class: "A", score: 85 },
     { class: "A", score: 90 },
     { class: "A", score: 68 },
     { class: "A", score: 95 },
     { class: "B", score: 80 },
     { class: "B", score: 75 },
     { class: "B", score: 92 },
     { class: "B", score: 88 },
     { class: "B", score: 70 }
   ]);
   const accumulatorResult = await students.aggregate([
     { $group: {
         _id: "$class",
         percentiles: { $percentile: { input: "$score", p: [0.25, 0.75], method:
"approximate" } }
       }}
   ]).toArray();
   console.log('Accumulator result:', accumulatorResult);

   // Expression usage: percentiles of an array field
   const surveys = db.collection('surveys');
   await surveys.insertMany([
     { _id: 1, responses: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20] },
     { _id: 2, responses: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19] }
   ]);
   const expressionResult = await surveys.aggregate([
     { $project: {
         quartiles: { $percentile: { input: "$responses", p: [0.25, 0.5, 0.75],
method: "approximate" } }
       }}
   ]).toArray();
   console.log('Expression result:', expressionResult);

   // Window operator usage: percentiles across each partition
   const sensorData = db.collection('sensorData');
```

```
    await sensorData.insertMany([
      { _id: 1, sensor: "A", reading: 72 },
      { _id: 2, sensor: "A", reading: 85 },
      { _id: 3, sensor: "A", reading: 90 },
      { _id: 4, sensor: "A", reading: 68 },
      { _id: 5, sensor: "A", reading: 95 },
      { _id: 6, sensor: "B", reading: 80 },
      { _id: 7, sensor: "B", reading: 75 },
      { _id: 8, sensor: "B", reading: 92 },
      { _id: 9, sensor: "B", reading: 88 },
      { _id: 10, sensor: "B", reading: 70 }
    ]);
    const windowResult = await sensorData.aggregate([
      {
        $setWindowFields: {
          partitionBy: "$sensor",
          output: {
            readingPercentiles: {
              $percentile: { input: "$reading", p: [0.25, 0.75], method:
 "approximate" },
              window: { documents: ["unbounded", "unbounded"] }
            }
          }
        }
      }
    ]).toArray();
    console.log('Window result:', windowResult);

  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
```

```
    try:
        db = client['test']

        # Accumulator usage: percentiles across grouped documents
        students = db['students']
        students.insert_many([
            { 'class': 'A', 'score': 72 },
            { 'class': 'A', 'score': 85 },
            { 'class': 'A', 'score': 90 },
            { 'class': 'A', 'score': 68 },
            { 'class': 'A', 'score': 95 },
            { 'class': 'B', 'score': 80 },
            { 'class': 'B', 'score': 75 },
            { 'class': 'B', 'score': 92 },
            { 'class': 'B', 'score': 88 },
            { 'class': 'B', 'score': 70 }
        ])
        accumulator_result = list(students.aggregate([
            { '$group': {
                '_id': '$class',
                'percentiles': { '$percentile': { 'input': '$score', 'p': [0.25,
0.75], 'method': 'approximate' } }
            }}
        ]))
        print('Accumulator result:', accumulator_result)

        # Expression usage: percentiles of an array field
        surveys = db['surveys']
        surveys.insert_many([
            { '_id': 1, 'responses': [2, 4, 6, 8, 10, 12, 14, 16, 18, 20] },
            { '_id': 2, 'responses': [1, 3, 5, 7, 9, 11, 13, 15, 17, 19] }
        ])
        expression_result = list(surveys.aggregate([
            { '$project': {
                'quartiles': { '$percentile': { 'input': '$responses', 'p': [0.25,
0.5, 0.75], 'method': 'approximate' } }
            }}
        ]))
        print('Expression result:', expression_result)

        # Window operator usage: percentiles across each partition
        sensor_data = db['sensorData']
        sensor_data.insert_many([
```

```
            { '_id': 1, 'sensor': 'A', 'reading': 72 },
            { '_id': 2, 'sensor': 'A', 'reading': 85 },
            { '_id': 3, 'sensor': 'A', 'reading': 90 },
            { '_id': 4, 'sensor': 'A', 'reading': 68 },
            { '_id': 5, 'sensor': 'A', 'reading': 95 },
            { '_id': 6, 'sensor': 'B', 'reading': 80 },
            { '_id': 7, 'sensor': 'B', 'reading': 75 },
            { '_id': 8, 'sensor': 'B', 'reading': 92 },
            { '_id': 9, 'sensor': 'B', 'reading': 88 },
            { '_id': 10, 'sensor': 'B', 'reading': 70 }
        ])
        window_result = list(sensor_data.aggregate([
            {
                '$setWindowFields': {
                    'partitionBy': '$sensor',
                    'output': {
                        'readingPercentiles': {
                            '$percentile': { 'input': '$reading', 'p': [0.25, 0.75],
 'method': 'approximate' },
                            'window': { 'documents': ['unbounded', 'unbounded'] }
                        }
                    }
                }
            }
        ]))
        print('Window result:', window_result)

    finally:
        client.close()

example()
```

$pow

Eingeführt in 8.0

Mit dem $pow Operator in Amazon DocumentDB können Sie eine Zahl potenzieren. Dies kann nützlich sein, um exponentielle Berechnungen innerhalb Ihrer Aggregationspipeline durchzuführen.

Parameter

- <number>(erforderlich): Die Zahl, die potenziert werden soll.

- `<exponent>`(erforderlich): Die Potenz, auf die die Zahl erhöht werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der $pow Operator verwendet wird, um das Quadrat einer Zahl zu berechnen.

Erstellen Sie Beispieldokumente

```
db.numbers.insertMany([
  { "_id": 1, "value": 2 },
  { "_id": 2, "value": 3 },
  { "_id": 3, "value": 4 }
]);
```

Beispiel für eine Abfrage

```
db.numbers.aggregate([
  { $addFields: { "square": { $pow: ["$value", 2] } } }
])
```

Ausgabe

```
[
  { "_id": 1, "value": 2, "square": 4 },
  { "_id": 2, "value": 3, "square": 9 },
  { "_id": 3, "value": 4, "square": 16 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des $pow Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

Hier ist ein Beispiel für die Verwendung des $pow-Operators in einer Node.js Anwendung:

```
const { MongoClient } = require('mongodb');
```

```
async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('numbers');

  const result = await collection.aggregate([
    { $addFields: { "square": { $pow: ["$value", 2] } } }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

Hier ist ein Beispiel für die Verwendung des $pow-Operators in einer Python-Anwendung:

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['numbers']

    result = list(collection.aggregate([
        { "$addFields": { "square": { "$pow": ["$value", 2] } } }
    ]))

    print(result)
    client.close()

example()
```

$push

Der `$push` Aggregationsoperator gibt ein Array aller Werte aus einem angegebenen Ausdruck für jede Gruppe zurück. Er wird in der Regel innerhalb der `$group` Phase verwendet, um Werte in einem Array zu akkumulieren.

Parameter

- `expression`: Der Ausdruck, der für jedes Dokument in der Gruppe ausgewertet werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$push` Operator verwendet wird, um alle Produktnamen für jede Kategorie zu sammeln.

Erstellen Sie Beispieldokumente

```
db.sales.insertMany([
  { _id: 1, category: "Electronics", product: "Laptop", amount: 1200 },
  { _id: 2, category: "Electronics", product: "Mouse", amount: 25 },
  { _id: 3, category: "Furniture", product: "Desk", amount: 350 },
  { _id: 4, category: "Furniture", product: "Chair", amount: 150 },
  { _id: 5, category: "Electronics", product: "Keyboard", amount: 75 }
]);
```

Beispiel für eine Abfrage

```
db.sales.aggregate([
  {
    $group: {
      _id: "$category",
      products: { $push: "$product" }
    }
  }
]);
```

Ausgabe

```
[
  { _id: 'Furniture', products: [ 'Desk', 'Chair' ] },
  { _id: 'Electronics', products: [ 'Laptop', 'Mouse', 'Keyboard' ] }
```

```
]
```

Beispiel für die Verwendung von Fensteroperatoren (MongoDB Shell)

Neu ab Version 8.0.2.

Der `$push` Operator kann auch als Fensterbediener auf der `$setWindowFields` Bühne verwendet werden. In diesem Zusammenhang gibt es ein Array mit allen Werten des angegebenen Ausdrucks für die Dokumente in jedem Fenster in der Sortierreihenfolge zurück. Sie geben den Operator unter dem `output` Feld an und definieren optional die Fenstergrenzen mit einem `window` Dokument.

> **Note**
>
> Bei Verwendung als Fensteroperator in `$setWindowFields` `$push` ist er auf 100 MB an Zwischendaten begrenzt. Eine Operation, die dieses Limit überschreitet, gibt einen Fehler zurück.

Erstellen Sie Beispieldokumente

```
db.purchases.insertMany([
  { _id: 1, category: "Electronics", day: 1, product: "Laptop" },
  { _id: 2, category: "Electronics", day: 2, product: "Mouse" },
  { _id: 3, category: "Electronics", day: 3, product: "Keyboard" },
  { _id: 4, category: "Furniture", day: 1, product: "Desk" },
  { _id: 5, category: "Furniture", day: 2, product: "Chair" }
]);
```

Beispiel für eine Abfrage

Das folgende Beispiel partitioniert die Dokumente nach`category`, sortiert jede Partition nach `day` und gibt eine fortlaufende Liste von `product` Werten vom Anfang der Partition bis zum aktuellen Dokument zurück.

```
db.purchases.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$category",
      sortBy: { day: 1 },
      output: {
        productsSoFar: {
```

```
          $push: "$product",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "category": "Electronics", "day": 1, "product": "Laptop",
 "productsSoFar": [ "Laptop" ] },
  { "_id": 2, "category": "Electronics", "day": 2, "product": "Mouse", "productsSoFar":
 [ "Laptop", "Mouse" ] },
  { "_id": 3, "category": "Electronics", "day": 3, "product": "Keyboard",
 "productsSoFar": [ "Laptop", "Mouse", "Keyboard" ] },
  { "_id": 4, "category": "Furniture", "day": 1, "product": "Desk", "productsSoFar":
 [ "Desk" ] },
  { "_id": 5, "category": "Furniture", "day": 2, "product": "Chair", "productsSoFar":
 [ "Desk", "Chair" ] }
]
```

Jedes Dokument wird um das Array aller `product` Werte innerhalb seiner Partition vom Anfang bis zum aktuellen Dokument in der Sortierreihenfolge erweitert. `productsSoFar`

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$push` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten. Die folgenden Beispiele zeigen sowohl die Verwendung des Akkumulators (in`$group`) als auch die Verwendung von Fensteroperatoren (in`$setWindowFields`):

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
```

```
// Accumulator usage: collect all values per group
const sales = db.collection('sales');
await sales.insertMany([
  { _id: 1, category: "Electronics", product: "Laptop", amount: 1200 },
  { _id: 2, category: "Electronics", product: "Mouse", amount: 25 },
  { _id: 3, category: "Furniture", product: "Desk", amount: 350 },
  { _id: 4, category: "Furniture", product: "Chair", amount: 150 },
  { _id: 5, category: "Electronics", product: "Keyboard", amount: 75 }
]);
const accumulatorResult = await sales.aggregate([
  {
    $group: {
      _id: "$category",
      products: { $push: "$product" }
    }
  }
]).toArray();
console.log('Accumulator result:', accumulatorResult);

// Window operator usage: running list of values within each partition
const purchases = db.collection('purchases');
await purchases.insertMany([
  { _id: 1, category: "Electronics", day: 1, product: "Laptop" },
  { _id: 2, category: "Electronics", day: 2, product: "Mouse" },
  { _id: 3, category: "Electronics", day: 3, product: "Keyboard" },
  { _id: 4, category: "Furniture", day: 1, product: "Desk" },
  { _id: 5, category: "Furniture", day: 2, product: "Chair" }
]);
const windowResult = await purchases.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$category",
      sortBy: { day: 1 },
      output: {
        productsSoFar: {
          $push: "$product",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]).toArray();
console.log('Window result:', windowResult);
```

```
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']

    # Accumulator usage: collect all values per group
    sales = db['sales']
    sales.insert_many([
        { '_id': 1, 'category': 'Electronics', 'product': 'Laptop', 'amount':
 1200 },
        { '_id': 2, 'category': 'Electronics', 'product': 'Mouse', 'amount': 25 },
        { '_id': 3, 'category': 'Furniture', 'product': 'Desk', 'amount': 350 },
        { '_id': 4, 'category': 'Furniture', 'product': 'Chair', 'amount': 150 },
        { '_id': 5, 'category': 'Electronics', 'product': 'Keyboard', 'amount': 75 }
    ])
    accumulator_result = list(sales.aggregate([
        {
            '$group': {
                '_id': '$category',
                'products': { '$push': '$product' }
            }
        }
    ]))
    print('Accumulator result:', accumulator_result)

    # Window operator usage: running list of values within each partition
    purchases = db['purchases']
    purchases.insert_many([
        { '_id': 1, 'category': 'Electronics', 'day': 1, 'product': 'Laptop' },
        { '_id': 2, 'category': 'Electronics', 'day': 2, 'product': 'Mouse' },
        { '_id': 3, 'category': 'Electronics', 'day': 3, 'product': 'Keyboard' },
        { '_id': 4, 'category': 'Furniture', 'day': 1, 'product': 'Desk' },
```

```
        { '_id': 5, 'category': 'Furniture', 'day': 2, 'product': 'Chair' }
    ])
    window_result = list(purchases.aggregate([
        {
            '$setWindowFields': {
                'partitionBy': '$category',
                'sortBy': { 'day': 1 },
                'output': {
                    'productsSoFar': {
                        '$push': '$product',
                        'window': { 'documents': ['unbounded', 'current'] }
                    }
                }
            }
        }
    ]))
    print('Window result:', window_result)

    client.close()

example()
```

$project

Mit dem `$project` Operator in Amazon DocumentDB können Sie Felder selektiv in Ausgabedokumente ein- oder ausschließen, Werte an die nächste Pipeline-Phase übergeben und neue Felder aus Eingabedokumentwerten berechnen.

Parameter

- `field`: Das Feld, das in die Ausgabedokumente ein- oder ausgeschlossen werden soll. Dabei kann es sich um einen Feldpfad handeln (z. B. „a.b.c“).
- `1`oder`true`: Schließt das Feld in die Ausgabe ein.
- `0`oder`false`: Schließt das Feld von der Ausgabe aus.

Beispiel (MongoDB Shell)

Das folgende Beispiel demonstriert die Verwendung des `$project` Operators bei der Erfassung von Studenten

Erstellen Sie Beispieldokumente

```
db.students.insertMany([
  { "_id": 1, "name": "Alejandro Rosalez", "math": 85, "science": 92, "grade": "A" },
  { "_id": 2, "name": "Carlos Salazar", "math": 78, "science": 84, "grade": "B" },
  { "_id": 3, "name": "Nikhil Jayashankar", "math": 95, "science": 89, "grade": "A" },
  { "_id": 4, "name": "Shirley Rodriguez", "math": 72, "science": 76, "grade": "B" }
  ]);
```

Diese Abfrage umfasst nur die `math` Felder `name` und in der Ausgabe. Das `_id` Feld ist standardmäßig enthalten, sofern es nicht ausdrücklich ausgeschlossen ist.

```
db.students.aggregate([
  { $project: { "name": 1, "math": 1 } }
])
```

Ausgabe

```
{ _id: 1, name: "Alejandro Rosalez", math: 85 }
{ _id: 2, name: "Carlos Salazar", math: 78 }
{ _id: 3, name: "Nikhil Jayashankar", math: 95 }
{ _id: 4, name: "Shirley Rodriguez", math: 72 }
```

Diese Abfrage schließt die `_id` Felder `grade` und aus der Ausgabe aus und zeigt alle anderen Felder (`name`,`math`,`science`) an.

```
db.students.aggregate([
  { $project: { "grade": 0, "_id": 0 } }
])
```

Ausgabe

```
{ name: "Alejandro Rosalez", math: 85, science: 92 }
{ name: "Carlos Salazar", math: 78, science: 84 }
{ name: "Nikhil Jayashankar", math: 95, science: 89 }
{ name: "Shirley Rodriguez", math: 72, science: 76 }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$project` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('students');

  const result = await collection.aggregate([
    { $project: { "name": 1, "math": 1 } }
  ]).toArray();
  console.log(result);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['students']

    result = list(collection.aggregate([
        { '$project': { 'name': 1, 'math': 1 } }
    ]))
    print(result)

    client.close()

example()
```

$ Radiant ToDegrees

Neu ab Version 8.0.1.

Der `$radiansToDegrees` Operator in Amazon DocumentDB konvertiert einen Wert von Bogenmaß in Grad.

Parameter

- `expression`: Ein Ausdruck, der in einen numerischen Wert im Bogenmaß aufgelöst wird.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$radiansToDegrees` Operator verwendet wird, um Bogenmaßwerte in Grad umzurechnen.

Erstellen Sie Beispieldokumente

```
db.angles.insertMany([
  { "_id": 1, "radians": 0 },
  { "_id": 2, "radians": 1.5707963267948966 },
  { "_id": 3, "radians": 3.141592653589793 },
  { "_id": 4, "radians": 6.283185307179586 }
]);
```

Beispiel für eine Abfrage

```
db.angles.aggregate([
  { $project: {
    "degrees": { $radiansToDegrees: "$radians" }
  }}
]);
```

Ausgabe

```
[
  { "_id": 1, "degrees": 0 },
  { "_id": 2, "degrees": 90 },
  { "_id": 3, "degrees": 180 },
  { "_id": 4, "degrees": 360 }
```

```
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$radiansToDegrees` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('angles');

  const result = await collection.aggregate([
    { $project: {
      "degrees": { $radiansToDegrees: "$radians" }
    }}
  ]).toArray();

  console.log(result);
  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['angles']

    result = list(collection.aggregate([
        { '$project': {
```

```
            'degrees': { '$radiansToDegrees': '$radians' }
        }}
    ]))

    print(result)
    client.close()

if __name__ == "__main__":
    main()
```

$ Rand

Neu ab Version 8.0

Der `$rand` Operator in Amazon DocumentDB wird verwendet, um eine Zufallszahl zwischen 0 und 1 zu generieren.

Parameter

Keine

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$rand` Operator verwendet wird, um nach dem Zufallsprinzip zwei Dokumente aus der `temp` Sammlung auszuwählen.

Erstellen Sie Beispieldokumente

```
db.items.insertMany([
  { "name": "pencil", "quantity": 110 },
  { "name": "pen", "quantity": 159 }
])
```

Beispiel für eine Abfrage

```
db.items.aggregate([
  {
    $project: {
      randomValue: { $rand: {} }
    }
  }
```

```
])
```

Ausgabe

```
[
  {
    _id: ObjectId('6924a5edd66dcae121d29517'),
    randomValue: 0.8615243955294392
  },
  {
    _id: ObjectId('6924a5edd66dcae121d29518'),
    randomValue: 0.22815483022099903
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$rand` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('items');

  const result = await collection.aggregate([
    {
      $project: {
        randomValue: { $rand: {} }
      }
    }
  ]).toArray();

  console.log(result);

  await client.close();
}
```

```
example();
```

Python

```
from pymongo import MongoClient

def example():
  client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
  db = client['test']
  collection = db['items']

  result = list(collection.aggregate([
      {
        "$project": {
          "randomValue": { "$rand": {} }
        }
      }
  ]))

  print(result)

  client.close()

example()
```

$range

Der `$range` Aggregationsoperator in Amazon DocumentDB wird verwendet, um ein Array von aufeinanderfolgenden Zahlen innerhalb eines bestimmten Bereichs zu erstellen. Dieser Operator ist besonders nützlich für die Generierung von Zahlenfolgen, wie z. B. Meilenmarkierungen für Verpflegungsstationen in einem Rennen, wie in den folgenden Beispielen gezeigt.

Parameter

- `start`: Der Startwert für den Bereich.
- `end`: Der Endwert für den Bereich.
- `step`: (optional) Der Schrittwert, der beim Generieren des Bereichs verwendet werden soll. Wenn nicht angegeben, ist der Standard-Schrittwert 1.

Beispiel (MongoDB Shell)

In diesem Beispiel verwenden wir den `$range` Operator, um die Meilenmarkierungen für Wasserstationen in einem Radrennen zu generieren.

Erstellen Sie Beispieldokumente

```
db.races.insertMany([
  { _id: 0, race: "STP", distance: 206 },
  { _id: 1, race: "RSVP", distance: 160 },
  { _id: 2, race: "Chilly Hilly", distance: 33 },
  { _id: 3, race: "Flying Wheels", distance: 100 }
]);
```

Beispiel für eine Abfrage

```
db.races.aggregate([
  {
    $project: {
      race: 1,
      "waterStations": { $range: [20, "$distance", 20] }
    }
  }
]);
```

Ausgabe

```
[
  {
    _id: 0,
    race: 'STP',
    waterStations: [
       20,  40,  60,  80,
      100, 120, 140, 160,
      180, 200
    ]
  },
  {
    _id: 1,
    race: 'RSVP',
    waterStations: [
       20,  40,  60, 80,
```

```
      100, 120, 140
    ]
  },
  { _id: 2, race: 'Chilly Hilly', waterStations: [ 20 ] },
  { _id: 3, race: 'Flying Wheels', waterStations: [ 20, 40, 60, 80 ] }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$range` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  try {
    await client.connect();
    const db = client.db('test');
    const collection = db.collection('races');

    const pipeline = [
      {
        $project: {
          race: 1,
          waterStations: { $range: [20, "$distance", 20] }
        }
      }
    ];

    const results = await collection.aggregate(pipeline).toArray();

    console.dir(results, { depth: null });

  } finally {
    await client.close();
  }
}
```

```
example().catch(console.error);
```

Python

```
from pymongo import MongoClient

def example():

  client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

  try:
      db = client.test
      collection = db.races

      pipeline = [
          {
              "$project": {
                  "race": 1,
                  "waterStations": { "$range": [20, "$distance", 20] }
              }
          }
      ]

      results = collection.aggregate(pipeline)

      for doc in results:
          print(doc)

  except Exception as e:
      print(f"An error occurred: {e}")

  finally:
      client.close()

example()
```

$ Rang

Neu ab Version 8.0.2.

Der `$rank` Fensteroperator in Amazon DocumentDB gibt die Position jedes Dokuments innerhalb seiner Partition entsprechend der Reihenfolge der `$setWindowFields` Phasen `sortBy` zurück. Dokumente mit demselben `sortBy` Wert erhalten denselben Rang, und beim nächsten eindeutigen Wert werden die dazwischenliegenden Positionen übersprungen, sodass Lücken entstehen (z. B.). `1, 2, 2, 4` Er wird unter dem `output` Feld der `$setWindowFields` Stufe angegeben.

Es ist ein Fensteroperator, der nur auf der `$setWindowFields` Bühne verwendet wird.

Parameter

- `$rank`nimmt ein leeres Dokument `{}` als Wert an.
- Es akzeptiert keinen `window` Frame.
- Es wird genau ein `sortBy` Schlüssel auf der `$setWindowFields` Bühne benötigt.

Beispiel (MongoDB Shell)

Das folgende Beispiel partitioniert die Dokumente nach`game`, sortiert jede Partition `score` in absteigender Reihenfolge und weist jedem Spieler einen Rang zu.

Erstellen Sie Beispieldokumente

```
db.players.insertMany([
  { _id: 1, game: "A", player: "Alice", score: 90 },
  { _id: 2, game: "A", player: "Bob", score: 85 },
  { _id: 3, game: "A", player: "Carol", score: 85 },
  { _id: 4, game: "A", player: "Dave", score: 70 },
  { _id: 5, game: "B", player: "Erin", score: 95 },
  { _id: 6, game: "B", player: "Frank", score: 80 }
]);
```

Beispiel für eine Abfrage

```
db.players.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$game",
      sortBy: { score: -1 },
      output: {
        rank: { $rank: {} }
      }
```

```
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "game": "A", "player": "Alice", "score": 90, "rank": 1 },
  { "_id": 2, "game": "A", "player": "Bob", "score": 85, "rank": 2 },
  { "_id": 3, "game": "A", "player": "Carol", "score": 85, "rank": 2 },
  { "_id": 4, "game": "A", "player": "Dave", "score": 70, "rank": 4 },
  { "_id": 5, "game": "B", "player": "Erin", "score": 95, "rank": 1 },
  { "_id": 6, "game": "B", "player": "Frank", "score": 80, "rank": 2 }
]
```

Innerhalb jeder game Partition werden die Spieler absteigend sortiert`score`. Bob und Carol liegen gleichauf 85 und teilen sich den Rang2, sodass der nächste Rang an Dave übergeht. 4

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$rank` Operators in der `$setWindowFields` Phase anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  try {
    await client.connect();
    const db = client.db('test');
    const players = db.collection('players');

    await players.insertMany([
      { _id: 1, game: "A", player: "Alice", score: 90 },
      { _id: 2, game: "A", player: "Bob", score: 85 },
      { _id: 3, game: "A", player: "Carol", score: 85 },
      { _id: 4, game: "A", player: "Dave", score: 70 },
```

```
      { _id: 5, game: "B", player: "Erin", score: 95 },
      { _id: 6, game: "B", player: "Frank", score: 80 }
    ]);

    const result = await players.aggregate([
      {
        $setWindowFields: {
          partitionBy: "$game",
          sortBy: { score: -1 },
          output: {
            rank: { $rank: {} }
          }
        }
      }
    ]).toArray();

    console.log(result);
  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    try:
        db = client['test']
        players = db['players']

        players.insert_many([
            { '_id': 1, 'game': 'A', 'player': 'Alice', 'score': 90 },
            { '_id': 2, 'game': 'A', 'player': 'Bob', 'score': 85 },
            { '_id': 3, 'game': 'A', 'player': 'Carol', 'score': 85 },
            { '_id': 4, 'game': 'A', 'player': 'Dave', 'score': 70 },
            { '_id': 5, 'game': 'B', 'player': 'Erin', 'score': 95 },
```

```
            { '_id': 6, 'game': 'B', 'player': 'Frank', 'score': 80 }
        ])

        result = list(players.aggregate([
            {
                '$setWindowFields': {
                    'partitionBy': '$game',
                    'sortBy': { 'score': -1 },
                    'output': {
                        'rank': { '$rank': {} }
                    }
                }
            }
        ]))

        print(result)
    finally:
        client.close()

example()
```

$redact

Die `$redact` Aggregationsphase in Amazon DocumentDB wird verwendet, um Inhalte anhand der Werte der angegebenen Felder selektiv in die Ausgabedokumente ein- oder auszuschließen. Dies ist besonders nützlich in Szenarien, in denen Sie die Sichtbarkeit sensibler Daten auf der Grundlage von Zugriffsebenen oder Benutzerberechtigungen kontrollieren müssen.

Parameter

- `$cond`: Ein Ausdruck, der entweder `$$KEEP``$$PRUNE`, oder `$$DESCEND` für jedes Feld im Dokument ausgewertet wird.
- `$$KEEP`: Behält das aktuelle Feld im Ausgabedokument bei.
- `$$PRUNE`: Entfernt das aktuelle Feld aus dem Ausgabedokument.
- `$$DESCEND`: Wendet den `$redact` Schritt rekursiv auf das aktuelle Feld an, bei dem es sich um ein Objekt oder ein Array handelt.

Beispiel (MongoDB Shell)

In diesem Beispiel verwenden wir die `$redact` Phase, um Bestellungen nach ihrem Status zu filtern und nur Bestellungen mit bestimmten Statuswerten anzuzeigen.

Erstellen Sie Beispieldokumente

```
db.orders.insert([
  { "_id": 1, "status": "shipped", "customer": "Carlos Salazar", "total": 150.00,
 "date": "2025-01-15" },
  { "_id": 2, "status": "processing", "customer": "Saanvi Sarkar", "total": 89.99,
 "date": "2025-01-20" },
  { "_id": 3, "status": "cancelled", "customer": "Zhang Wei", "total": 220.50, "date":
 "2025-01-18" }
])
```

Beispiel für eine Abfrage

```
db.orders.aggregate([
  {
    $redact: {
      $cond: {
        if: { $in: ["$status", ["shipped", "processing"]] },
        then: "$$KEEP",
        else: "$$PRUNE"
      }
    }
  }
])
```

Ausgabe

```
[
  { _id: 1, status: 'shipped', customer: 'Carlos Salazar', total: 150, date:
 '2025-01-15' },
  { _id: 2, status: 'processing', customer: 'Saanvi Sarkar', total: 89.99, date:
 '2025-01-20' }
]
```

In diesem Beispiel überprüft die `$redact` Phase den Wert des `status` Felds in jedem Dokument. Wenn `status` es sich um „Versand“ oder „Bearbeitung“ handelt, wird das Dokument aufbewahrt (`$$KEEP`). Andernfalls wird das Dokument beschnitten ()`$$PRUNE`.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$redact` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('orders');

  const result = await collection.aggregate([
    {
      $redact: {
        $cond: {
          if: { $in: ['$status', ['shipped', 'processing']] },
          then: '$$KEEP',
          else: '$$PRUNE'
        }
      }
    }
  ]).toArray();

  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
```

```
    collection = db['orders']

    result = list(collection.aggregate([
        {
            '$redact': {
                '$cond': {
                    'if': { '$in': ['$status', ['shipped', 'processing']] },
                    'then': '$$KEEP',
                    'else': '$$PRUNE'
                }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$reduce

Der `$reduce` Aggregationsoperator in Amazon DocumentDB wird verwendet, um eine Funktion mit zwei Argumenten kumulativ auf die Elemente eines Arrays anzuwenden, um das Array auf einen einzigen Wert zu reduzieren. Dieser Operator ist besonders nützlich für die Durchführung komplexer Berechnungen oder Transformationen von Array-Daten innerhalb der Aggregationspipeline.

Parameter

- `input`: Das Array, das reduziert werden soll.
- `initialValue`: Der Anfangswert, der bei der Reduktionsoperation verwendet werden soll.
- `in`: Der Ausdruck, der für jedes Element des `input` Arrays ausgewertet werden soll. Dieser Ausdruck sollte einen Wert zurückgeben, der in der nächsten Iteration der Reduktion verwendet wird.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$reduce` Operator verwendet wird, um die Summe aller Elemente in einem Array zu berechnen.

Erstellen Sie Beispieldokumente

```
db.orders.insertMany([
  { "_id": 1, "items": [1, 2, 3, 4, 5] },
  { "_id": 2, "items": [10, 20, 30] },
  { "_id": 3, "items": [5, 15, 25, 35] },
  { "_id": 4, "items": [100, 200] }
])
```

Beispiel für eine Abfrage

```
db.orders.aggregate([
  {
    $project: {
      total: {
        $reduce: {
          input: "$items",
          initialValue: 0,
          in: { $add: ["$$value", "$$this"] }
        }
      }
    }
  }
])
```

Ausgabe

```
[
  { "_id": 1, "total": 15 },
  { "_id": 2, "total": 60 },
  { "_id": 3, "total": 80 },
  { "_id": 4, "total": 300 }
]
```

Der `$reduce` Operator iteriert über das `items` Array und fügt jedes Element dem `initialValue` Wert 0 hinzu. Das Ergebnis ist die Summe aller Elemente im Array.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$reduce` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

Hier ist ein Beispiel für die Verwendung des `$reduce` Operators in einer Node.js Anwendung:

```
const { MongoClient } = require("mongodb");

async function main() {
  const client = await MongoClient.connect("mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false");
  const db = client.db("test");
  const orders = db.collection("orders");

  const result = await orders.aggregate([
    {
      $project: {
        total: {
          $reduce: {
            input: "$items",
            initialValue: 0,
            in: { $add: ["$$value", "$$this"] }
          }
        }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

main();
```

Python

Hier ist ein Beispiel für die Verwendung des `$reduce` Operators in einer Python-Anwendung:

```
from pymongo import MongoClient

def main():
    client = MongoClient("mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false")
```

```
    db = client["test"]
    orders = db["orders"]

    result = list(orders.aggregate([
        {
            "$project": {
                "total": {
                    "$reduce": {
                        "input": "$items",
                        "initialValue": 0,
                        "in": { "$add": ["$$value", "$$this"] }
                    }
                }
            }
        }
    ]))

    print(result)
    client.close()

if __name__ == "__main__":
    main()
```

$RegexFind

Neu ab Version 5.0.

Wird von Elastic Cluster nicht unterstützt.

Der `$regexFind` Operator in Amazon DocumentDB wird verwendet, um den Abgleich regulärer Ausdrücke für Zeichenkettenfelder in Dokumenten durchzuführen. Es ermöglicht Ihnen, nach bestimmten Teilzeichenfolgen zu suchen und diese zu extrahieren, die einem bestimmten Muster für reguläre Ausdrücke entsprechen.

Parameter

- `input`: Das Zeichenkettenfeld oder der Ausdruck, nach dem gesucht werden soll.
- `regex`: Das Muster für reguläre Ausdrücke, nach dem gesucht werden soll.
- `options`: (optional) Ein Objekt, das optionale Parameter für den regulären Ausdruck angibt, z. B. Berücksichtigung von Groß- und Kleinschreibung und mehrzeiliger Abgleich. Unterstützte Optionen sind `i` (ohne Berücksichtigung der Groß-/Kleinschreibung) und `m` (mehrzeilig).

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$regexFind` Operator verwendet wird, um nach Dokumenten zu suchen, bei denen das `name` Feld einem bestimmten Muster für reguläre Ausdrücke entspricht.

Erstellen Sie Beispieldokumente

```
db.users.insertMany([
  { "_id": 1, name: "John Doe", email: "john@example.com" },
  { "_id": 2, name: "Diego Ramirez", email: "diego@example.com" },
  { "_id": 3, name: "Alejandro Rosalez", email: "alejandro@example.com" },
  { "_id": 4, name: "Shirley Rodriguez", email: "shirley@example.com" }
]);
```

Beispiel für eine Abfrage

```
db.users.aggregate([
  {
    $project: {
      names: {
        $regexFind: { input: '$name', regex: 'j', options: 'i' }
      }
    }
  },
  { $match: {names: {$ne: null}}}
])
```

Diese Abfrage gibt alle Dokumente zurück, in denen das `name` Feld den Buchstaben „j“ enthält (Groß- und Kleinschreibung wird nicht beachtet).

Ausgabe

```
[
  { _id: 1, names: { match: 'J', idx: 0, captures: [] } }
]
```

Hinweis: Wenn Ihre Abfrage Amazon DocumentDB Planner Version 1 verwendet, müssen Sie einen Hinweis verwenden, um einen Index zu verwenden. Ohne einen Hinweis führt die Abfrage möglicherweise einen Sammlungsscan durch. Informationen zu Ihrer Planer-Version und weitere Informationen zur Verwendung von Hinweisen finden Sie unterAbfrageplaner v2.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$regexFind` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const users = db.collection('users');

  const results = await users.aggregate([
    { $project: { names: { $regexFind: { input: "$name", regex: "john", options:
 "i" }}}},
    { $match: {names: {$ne: null}}}
  ]).toArray();


  console.log(results);

  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
db = client['test']
users = db['users']

results = list(users.aggregate([
    { "$project": { "names": { "$regexFind": { "input": "$name", "regex": "john",
 "options": "i" }}}},
```

```
    { "$match": {"names": {"$ne": None}}}
]))

print(results)

client.close()
```

$ regulärer Ausdruck FindAll

In 5.0 eingeführt

Der `$regexFindAll` Operator in Amazon DocumentDB wird verwendet, um den Abgleich regulärer Ausdrücke für Zeichenkettenfelder in Dokumenten durchzuführen. Es ermöglicht Ihnen, nach bestimmten Teilzeichenfolgen zu suchen und diese zu extrahieren, die einem bestimmten Muster für reguläre Ausdrücke entsprechen. Dabei werden alle Treffer des regulären Ausdrucks zurückgegeben.

Parameter

- `input`: Das Zeichenkettenfeld oder der Ausdruck, nach dem gesucht werden soll.
- `regex`: Das Muster für reguläre Ausdrücke, nach dem gesucht werden soll.
- `options`: (optional) Ein Objekt, das optionale Parameter für den regulären Ausdruck angibt, z. B. Berücksichtigung von Groß- und Kleinschreibung und mehrzeiliger Abgleich. Unterstützte Optionen sind `i` (ohne Berücksichtigung der Groß-/Kleinschreibung) und `m` (mehrzeilig).

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$regexFindAll` Operator verwendet wird, um alle Buchstabenfolgen aus dem `email` Feld zu extrahieren.

Erstellen Sie Beispieldokumente

```
db.users.insertMany([
  { _id: 1, name: "John Doe", email: "john@example.com", phone: "555-1234" },
  { _id: 2, name: "Jane Roe", email: "jane@example.com", phone: "555-5678" },
  { _id: 3, name: "Carlos Salazar", email: "carlos@example.com", phone: "555-3456" },
  { _id: 4, name: "Saanvi Sarkar", email: "saanvi@example.com", phone: "555-7890" }

]);
```

Beispiel für eine Abfrage

```
db.users.aggregate([
  {
    $project: {
      name: 1,
      emailMatches: {
        $regexFindAll: { input: '$email', regex: '[a-z]+', options: 'i' }
      }
    }
  }
])
```

Ausgabe

```
[
  {
    _id: 1,
    name: 'John Doe',
    emailMatches: [
      { match: 'john', idx: 0, captures: [] },
      { match: 'example', idx: 5, captures: [] },
      { match: 'com', idx: 13, captures: [] }
    ]
  },
  {
    _id: 2,
    name: 'Jane Roe',
    emailMatches: [
      { match: 'jane', idx: 0, captures: [] },
      { match: 'example', idx: 5, captures: [] },
      { match: 'com', idx: 13, captures: [] }
    ]
  },
  {
    _id: 3,
    name: 'Carlos Salazar',
    emailMatches: [
      { match: 'carlos', idx: 0, captures: [] },
      { match: 'example', idx: 7, captures: [] },
      { match: 'com', idx: 15, captures: [] }
    ]
  },
  {
    _id: 4,
```

```
    name: 'Saanvi Sarkar',
    emailMatches: [
      { match: 'saanvi', idx: 0, captures: [] },
      { match: 'example', idx: 7, captures: [] },
      { match: 'com', idx: 15, captures: [] }
    ]
  }
]
```

Hinweis: Wenn Ihre Abfrage Amazon DocumentDB Planner Version 1 verwendet, müssen Sie einen Hinweis verwenden, um einen Index zu verwenden. Ohne einen Hinweis führt die Abfrage möglicherweise einen Sammlungsscan durch. Informationen zu Ihrer Planer-Version und weitere Informationen zur Verwendung von Hinweisen finden Sie unterAbfrageplaner v2.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$regexFindAll` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

Hier ist ein Beispiel für die Verwendung des `$regexFind` Operators in einer Node.js Anwendung:

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const users = db.collection('users');

  const results = await users.aggregate([
    {
      $project: {
        name: 1,
        emailMatches: {
          $regexFindAll: { input: "$email", regex: "[a-z]+", options: "i" }
        }
      }
    }
  ]).toArray();
```

```
  console.log(JSON.stringify(results, null, 2));

  await client.close();
}

main();
```

Python

Hier ist ein Beispiel für die Verwendung des `$regexFind` Operators in einer Python-Anwendung:

```
from pymongo import MongoClient

client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
db = client['test']
users = db['users']

results = list(users.aggregate([
    {
        "$project": {
            "name": 1,
            "emailMatches": {
                "$regexFindAll": {
                    "input": "$email",
                    "regex": "[a-z]+",
                    "options": "i"
                }
            }
        }
    }
]))

print(results)

client.close()
```

$RegexMatch

Neu ab Version 5.0. Wird von Elastic Cluster nicht unterstützt.

Der `$regexMatch` Operator in Amazon DocumentDB wird verwendet, um den Abgleich regulärer Ausdrücke für Zeichenkettenfelder durchzuführen. Er gibt einen booleschen Wert (`true`oder`false`) zurück, der angibt, ob die Eingabezeichenfolge dem angegebenen Muster entspricht.

Parameter

- `input`: Die Zeichenfolge, die anhand des regulären Ausdrucks getestet werden soll.
- `regex`: Das Muster für reguläre Ausdrücke, das abgeglichen werden soll.
- `options`: (Optional) Markierungen zur Änderung des Verhaltens regulärer Ausdrücke, z. B. beim Abgleich ohne Berücksichtigung der Groß- und Kleinschreibung (`i`) oder beim mehrzeiligen Abgleich (). `m`

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$regexMatch` Operator verwendet wird, um zu überprüfen, ob Namen mit dem Buchstaben 'M' beginnen. Der Operator gibt `false` für jedes `true` Dokument oder zurück.

Erstellen Sie Beispieldokumente

```
db.users.insertMany([
  { "_id":1, name: "María García", email: "maría@example.com" },
  { "_id":2, name: "Arnav Desai", email: "arnav@example.com" },
  { "_id":3, name: "Martha Rivera", email: "martha@example.com" },
  { "_id":4, name: "Richard Roe", email: "richard@example.com" },

]);
```

Beispiel für eine Abfrage

```
db.users.aggregate([
  {
    $project: {
      name: 1,
      startsWithM: {
        $regexMatch: {
          input: "$name",
          regex: "^M",
          options: "i"
        }
```

```
      }
    }
  }
]);
```

Ausgabe

```
{ _id: 1, name: 'María García', startsWithM: true },
{ _id: 2, name: 'Arnav Desai', startsWithM: false },
{ _id: 3, name: 'Martha Rivera', startsWithM: true },
{ _id: 4, name: 'Richard Roe', startsWithM: false }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$regexMatch` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function checkNamePattern() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('users');

  const result = await collection.aggregate([
    {
      $project: {
        name: 1,
        startsWithM: {
          $regexMatch: {
            input: "$name",
            regex: "^M",
            options: "i"
          }
        }
      }
    }
  ]).toArray();
```

```
  console.log(result);

  await client.close();
}

checkNamePattern();
```

Python

```
from pymongo import MongoClient

def check_name_pattern():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.users

    result = list(collection.aggregate([
        {
            '$project': {
                'name': 1,
                'startsWithM': {
                    '$regexMatch': {
                        'input': '$name',
                        'regex': '^M',
                        'options': 'i'
                    }
                }
            }
        }
    ]))

    print(result)

    client.close()

check_name_pattern()
```

$ Alles ersetzen

In 5.0 eingeführt

Der `$replaceAll` Operator in Amazon DocumentDB wird verwendet, um alle Vorkommen eines bestimmten Zeichenkettenmusters in einem Feld durch eine neue Zeichenfolge zu ersetzen. Dieser Operator kann für Aufgaben wie Datennormalisierung, Textreinigung und Zeichenkettenmanipulation nützlich sein.

Parameter

- `input`: Das Feld oder der Ausdruck, der die zu ersetzende Zeichenfolge enthält.
- `find`: Das Zeichenkettenmuster, nach dem gesucht und ersetzt werden soll.
- `replacement`: Die Zeichenfolge, durch die die übereinstimmenden Vorkommen ersetzt werden sollen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$replaceAll` Operator in einer Aggregationspipeline verwendet wird, um alle Vorkommen der Zeichenfolge „Chocolatier“ durch „Chocolate Co“ zu ersetzen. im Feld „BrandName“ einer Kollektion „Produkte“.

Erstellen Sie Beispieldokumente

```
db.products.insertMany([
  {
    "_id": 1,
    "productId": "PROD-0Y9GL0",
    "brandName": "Gordon's Chocolatier",
    "category": "CPG",
    "rating": {
      "average": 4.8
    }
  },
  {
    "_id": 2,
    "productId": "PROD-1X2YZ3",
    "brandName": "Premium Chocolatier",
    "category": "CPG",
    "rating": {
      "average": 4.5
    }
  },
  {
    "_id": 3,
```

```
    "productId": "PROD-Y2E9H5",
    "name": "Nutrition Co. - Original Corn Flakes Cereal",
    "category": "Breakfast Cereals",
    "price": 8.5
  }
]);
```

Beispiel für eine Abfrage

```
db.products.aggregate([
  {
    $addFields: {
      "brandName": {
        $replaceAll: {
          input: "$brandName",
          find: "Chocolatier",
          replacement: "Chocolate Co."
        }
      }
    }
  }
])
```

Ausgabe

```
[
  {
    _id: 1,
    productId: 'PROD-0Y9GL0',
    brandName: "Gordon's Chocolate Co.",
    category: 'CPG',
    rating: { average: 4.8 }
  },
  {
    _id: 2,
    productId: 'PROD-1X2YZ3',
    brandName: 'Premium Chocolate Co.',
    category: 'CPG',
    rating: { average: 4.5 }
  },
  {
    _id: 3,
    productId: 'PROD-Y2E9H5',
```

```
    name: 'Nutrition Co. - Original Corn Flakes Cereal',
    category: 'Breakfast Cereals',
    price: 8.5,
    brandName: null
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$replaceAll` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function replaceAll() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('products');

  const results = await collection.aggregate([
    {
      $addFields: {
        "brandName": {
          $replaceAll: {
            input: "$brandName",
            find: "Chocolatier",
            replacement: "Chocolate Co."
          }
        }
      }
    }
  ]).toArray();

  console.log(results);

  await client.close();
}

replaceAll();
```

Python

```
from pymongo import MongoClient

def replace_all():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.products

    results = list(collection.aggregate([
        {
            "$addFields": {
                "brandName": {
                    "$replaceAll": {
                        "input": "$brandName",
                        "find": "Chocolatier",
                        "replacement": "Chocolate Co."
                    }
                }
            }
        }
    ]))

    print(results)

    client.close()

replace_all()
```

$ ersetzenEins

In 5.0 eingeführt

Der `$replaceOne` Operator in Amazon DocumentDB ist ein Operator für Zeichenkettenausdrücke, der innerhalb von Aggregationspipelines verwendet wird, um das erste Vorkommen einer angegebenen Teilzeichenfolge innerhalb einer Zeichenfolge durch eine Ersatzzeichenfolge zu ersetzen. Dieser Operator unterscheidet Groß- und Kleinschreibung und ersetzt nur die erste gefundene Übereinstimmung.

Parameter

- `input`: Die Zeichenfolge (das Feld), nach der die Suche durchgeführt werden soll.
- `find`: Die Zeichenfolge, nach der in der Eingabe gesucht werden soll.
- `replacement`: Die Zeichenfolge, die das erste Vorkommen des Suchbegriffs in der Eingabe (Feld) ersetzen soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$replaceOne` Operator innerhalb einer Aggregationspipeline verwendet wird, um Teilzeichenfolgen in Produktnamen zu ersetzen.

Erstellen Sie Beispieldokumente

```
db.products.insertMany([
  { "_id":1, "productId": "PROD-0Y9GL0", "name": "Gordon's Extra Creamy Milk Chocolate
 - Pack of 4", "category": "Confectionery", "price": 24.99 },
  { "_id":2, "productId": "PROD-Y2E9H5", "name": "Nutrition Co. - Original Corn Flakes
 Cereal", "category": "Breakfast Cereals", "price": 8.50 },
  { "_id":3, "productId": "PROD-Z3F8K2", "name": "Gordon's Dark Chocolate (90% Cocoa)
 Pack - Pack of 4", "category": "Confectionery", "price": 28.99 }
]);
```

Beispiel für eine Aggregation

```
db.products.aggregate([
  {
    $addFields: {
      standardizedName: {
        $replaceOne: {
          input: "$name",
          find: "Pack",
          replacement: "Package"
        }
      }
    }
  }
]);
```

Ausgabe

Die Ausgabe zeigt, dass nur das erste Vorkommen von „Pack“ in jedem Produktnamen durch „Package“ ersetzt wurde.

```
[
  {
    _id: 1,
    productId: 'PROD-0Y9GL0',
    name: "Gordon's Extra Creamy Milk Chocolate - Pack of 4",
    category: 'Confectionery',
    price: 24.99,
    standardizedName: "Gordon's Extra Creamy Milk Chocolate - Package of 4"
  },
  {
    _id: 2,
    productId: 'PROD-Y2E9H5',
    name: 'Nutrition Co. - Original Corn Flakes Cereal',
    category: 'Breakfast Cereals',
    price: 8.5,
    standardizedName: 'Nutrition Co. - Original Corn Flakes Cereal'
  },
  {
    _id: 3,
    productId: 'PROD-Z3F8K2',
    name: "Gordon's Dark Chocolate (90% Cocoa) Pack - Pack of 4",
    category: 'Confectionery',
    price: 28.99,
    standardizedName: "Gordon's Dark Chocolate (90% Cocoa) Package - Pack of 4"
  }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$replaceOne` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function replaceOne() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('products');

  const pipeline = [
```

```
    {
      $addFields: {
        standardizedName: {
          $replaceOne: {
            input: '$name',
            find: 'Pack',
            replacement: 'Package'
          }
        }
      }
    }
  ];

  const result = await collection.aggregate(pipeline).toArray();
  console.log(result);

  await client.close();
}

replaceOne();
```

Python

```
from pymongo import MongoClient

def replaceOne():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['products']

    pipeline = [
        {
            '$addFields': {
                'standardizedName': {
                    '$replaceOne': {
                        'input': '$name',
                        'find': 'Pack',
                        'replacement': 'Package'
                    }
                }
            }
```

```
        }
    ]

    result = list(collection.aggregate(pipeline))
    print(result)

    client.close()

replaceOne()
```

$replaceRoot

Der `$replaceRoot` Operator wird verwendet, um das Stammdokument durch das angegebene eingebettete Dokument zu ersetzen. Dies ist nützlich, wenn Sie ein verschachteltes Dokument auf die oberste Ebene hochstufen oder Ihre Datenausgabe neu strukturieren möchten.

Parameter

- `newRoot`: Das neue Stammdokument, das das bestehende Stammdokument ersetzen wird.

Beispiel (MongoDB Shell)

Dieses Beispiel zeigt, wie Lieferadressinformationen aus Kundenbestellungen extrahiert werden, was für die Generierung von Versandetiketten oder Adresslisten nützlich ist.

Erstellen Sie Beispieldokumente

```
db.orders.insertMany([
  {
    "_id":1, "orderId": "ORD-2024-001", "customerId": "CUST-12345", "orderDate":
 "2024-01-15", "shippingAddress": { "name": "María García", "street": "123
 Main St", "city": "Seattle", "state": "WA", "zipCode": "98101", "country":
 "USA" },"totalAmount": 149.99 },
  { "_id":2, "orderId": "ORD-2024-002", "customerId": "CUST-67890", "orderDate":
 "2024-01-16", "shippingAddress": { "name": "Arnav Desai", "street": "456 Oak
 Ave", "city": "Portland", "state": "OR", "zipCode": "97201", "country": "USA" },
 "totalAmount": 89.50 } ])
```

Beispiel für eine Abfrage

```
db.orders.aggregate([
```

```
  {
    $replaceRoot: {
      newRoot: "$shippingAddress"
    }
  }
])
```

Ausgabe

```
{
    name: 'María García',
    street: '123 Main St',
    city: 'Seattle',
    state: 'WA',
    zipCode: '98101',
    country: 'USA'
  },
  {
    name: 'Arnav Desai',
    street: '456 Oak Ave',
    city: 'Portland',
    state: 'OR',
    zipCode: '97201',
    country: 'USA'
  }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$replaceRoot` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function extractShippingAddresses() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('orders');

  const result = await collection.aggregate([
```

```
    {
      $replaceRoot: {
        newRoot: "$shippingAddress"
      }
    }
  ]).toArray();

  console.log(result);

  await client.close();
}

extractShippingAddresses();
```

Python

```
from pymongo import MongoClient

def extract_shipping_addresses():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['orders']

    result = list(collection.aggregate([
        {
            "$replaceRoot": {
                "newRoot": "$shippingAddress"
            }
        }
    ]))

    print(result)

    client.close()

extract_shipping_addresses()
```

$ ersetzendurch

Neu ab Version 8.0

Wird von Elastic Cluster nicht unterstützt.

Die `$replaceWith` Aggregationsphase in Amazon DocumentDB wird verwendet, um das Eingabedokument durch ein neues Dokument zu ersetzen. Alle vorhandenen Felder im Eingabedokument, einschließlich des Felds _id, werden durch das neue Dokument ersetzt. `$replaceWith`wird häufig verwendet, um Dokumente zu vereinfachen oder ein eingebettetes Dokument auf die oberste Ebene heraufzustufen.

Parameter

- `<replacement>`(erforderlich): Das neue Dokument, das das bestehende Dokument ersetzen wird.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$replaceWith` Operator verwendet wird, um ein vorhandenes Dokument in einer Amazon DocumentDB-Sammlung zu ersetzen.

Erstellen Sie Beispieldokumente

```
db.restaurants.insertMany([
  {
    "restaurantId": "REST-0Y9GL0",
    "name": "Biryani Adda",
    "cuisine": "Indian",
    "ratings": [ 3, 4, 3, 2, 2, 4, 1, 5, 5, 5 ]
  },
  {
    "restaurantId": "REST-8L2PX9",
    "name": "The Burger Spot",
    "cuisine": "American",
    "ratings": [ 2, 3, 4, 5, 3, 1, 1, 2, 4 ]
  }
]);
```

Beispiel für eine Abfrage

```
db.restaurants.aggregate([
  { $replaceWith: {
      name: "$name",
      cuisine: "$cuisine",
      rating: { $avg: "$ratings" }
```

```
    }
  }
]);
```

Ausgabe

```
[
  {
    name: 'Biryani Adda',
    cuisine: 'Indian',
    rating: 3.4
  },
  {
    name: 'The Burger Spot',
    cuisine: 'American',
    rating: 2.7777777777777777
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$replaceWith` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function replaceDoc() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('restaurants');

  const result = await collection.aggregate([
    { $replaceWith: {
        name: "description_index",
        cuisine: 2,
        rating: { $avg: "$ratings" }
      }
    }
  ]).toArray();
```

```
  console.log(result);
  client.close();
}

replaceDoc();
```

Python

```
from pymongo import MongoClient


def replace_document():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.restaurants

    result = list(collection.aggregate([
        {
            '$replaceWith': {
                'name': "$name",
                'cuisine': "$cuisine",
                'rating': { '$avg': "$ratings"}
            }
        }
    ]))

    print(result)
    client.close()

replace_document()
```

$reverseArray

Der `$reverseArray` Operator in Amazon DocumentDB wird verwendet, um die Elemente eines Arrays in der angegebenen Reihenfolge umzukehren. Dieser Operator ist nützlich, wenn Sie die Elemente eines Arrays in umgekehrter Richtung neu anordnen müssen.

Parameter

- `expression`: Der Array-Ausdruck, der umgekehrt werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$reverseArray` Operator verwendet wird, um die Reihenfolge der Elemente in einem Array umzukehren.

Erstellen Sie Beispieldokumente

```
db.miles.insertMany([
  { "_id" : 1, "member_since" : ISODate("1987-01-01T00:00:00Z"), "credit_card" : false,
 "flight_miles" : [ 1205, 2560, 880 ]},
  { "_id" : 2, "member_since" : ISODate("1982-01-01T00:00:00Z"), "credit_card" : true,
 "flight_miles" : [ 1205, 2560, 890, 2780]},
  { "_id" : 3, "member_since" : ISODate("1999-01-01T00:00:00Z"), "credit_card" : true,
 "flight_miles" : [ 1205, 880]}
]);
```

Beispiel für eine Abfrage

```
db.miles.aggregate([
  {
    $project: {
      _id: 1,
      member_since: 1,
      credit_card: 1,
      reversed_flight_miles: { $reverseArray: "$flight_miles" }
    }
  }
]);
```

Ausgabe

```
{ "_id" : 1, "member_since" : ISODate("1987-01-01T00:00:00Z"), "credit_card" : false,
 "reversed_flight_miles" : [ 880, 2560, 1205 ] }
{ "_id" : 2, "member_since" : ISODate("1982-01-01T00:00:00Z"), "credit_card" : true,
 "reversed_flight_miles" : [ 2780, 890, 2560, 1205 ] }
{ "_id" : 3, "member_since" : ISODate("1999-01-01T00:00:00Z"), "credit_card" : true,
 "reversed_flight_miles" : [ 880, 1205 ] }
```

In diesem Beispiel wird der `$reverseArray` Operator verwendet, um die Reihenfolge des `flight_miles` Arrays umzukehren. Das resultierende `reversed_flight_miles` Feld in der Ausgabe zeigt die Elemente des Arrays in umgekehrter Reihenfolge.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$reverseArray` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

Hier ist ein Beispiel für die Verwendung des `$reverseArray` Operators in einer Node.js Anwendung:

```
const { MongoClient } = require('mongodb');

async function reverseArray() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('miles');

  const result = await collection.aggregate([
    {
      $project: {
        _id: 1,
        member_since: 1,
        credit_card: 1,
        reversed_flight_miles: { $reverseArray: '$flight_miles' }
      }
    }
  ]).toArray();

  console.log(result);
  client.close();
}

reverseArray();
```

Python

Hier ist ein Beispiel für die Verwendung des `$reverseArray` Operators in einer Python-Anwendung:

```
from pymongo import MongoClient

def reverse_array():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.miles

    result = list(collection.aggregate([
        {
            '$project': {
                '_id': 1,
                'member_since': 1,
                'credit_card': 1,
                'reversed_flight_miles': { '$reverseArray': '$flight_miles' }
            }
        }
    ]))

    print(result)
    client.close()

reverse_array()
```

$ Runde

Neu ab Version 8.0.1.

Der `$round` Operator in Amazon DocumentDB rundet eine Zahl auf eine angegebene Dezimalstelle. Wenn ein Wert genau in der Mitte zwischen den beiden nächstgelegenen Werten an dieser Stelle liegt, wird auf die nächste gerade Stelle `$round` gerundet. Beispiel: 2,5 Runden auf 2 und 3,5 Runden auf 4.

Parameter

- `number`: Ein Ausdruck, der in eine Zahl aufgelöst wird.

- `place`: Optional. Ein Ganzzahlausdruck zwischen -20 und 100, der die Anzahl der Dezimalstellen angibt, auf die gerundet werden soll. Ein negativer Wert wird auf die linke Seite der Dezimalstelle gerundet. Der Standardwert ist 0.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$round` Operator verwendet wird, um numerische Werte auf eine Dezimalstelle zu runden.

Rundung mit einem positiven Stellenwert

```
db.measurements.insertMany([
  {_id: 1, value: 3.456},
  {_id: 2, value: 7.891},
  {_id: 3, value: 12.345}
]);
```

Beispiel für eine Abfrage

```
db.measurements.aggregate([
  { $project: { rounded: { $round: ["$value", 1] } } }
]);
```

Ausgabe

```
[
  {_id: 1, rounded: 3.5},
  {_id: 2, rounded: 7.9},
  {_id: 3, rounded: 12.3}
]
```

Auf halber Strecke liegende Werte auf die nächste gerade Ziffer runden

Wenn ein Wert genau in der Mitte zwischen den beiden nächstgelegenen Werten an der Zielstelle liegt, wird auf die nächste gerade Ziffer gerundet, anstatt immer `$round` aufzurunden. Im folgenden Beispiel werden Werte auf halber Strecke auf ganze ganze Zahlen gerundet.

```
db.halfway.insertMany([
  {_id: 1, value: 10.5},
  {_id: 2, value: 11.5},
```

```
  {_id: 3, value: 12.5},
  {_id: 4, value: 13.5}
]);
```

```
db.halfway.aggregate([
  { $project: { rounded: { $round: ["$value", 0] } } }
]);
```

Ausgabe

```
[
  {_id: 1, rounded: 10},
  {_id: 2, rounded: 12},
  {_id: 3, rounded: 12},
  {_id: 4, rounded: 14}
]
```

Weil genau in der Mitte zwischen `10` und `10.5` liegt`11`, wird auf den nächsten geraden Wert gerundet. `10` Ähnlich, `11.5` und `12.5` beide werden auf den nächsten geraden Wert `12` `13.5` gerundet und auf `14` gerundet.

Rundung mit einem negativen Stellenwert

Wenn negativ `place` ist, wird auf die linke Seite der Dezimalstelle `$round` gerundet. Im folgenden Beispiel werden Werte mit der ersten Ziffer links von der Dezimalstelle gerundet.

```
db.samples.insertMany([
  {_id: 1, value: 19.25},
  {_id: 2, value: 28.73},
  {_id: 3, value: 34.32}
]);
```

```
db.samples.aggregate([
  { $project: { rounded: { $round: ["$value", -1] } } }
]);
```

Ausgabe

```
[
  {_id: 1, rounded: 20},
```

```
  {_id: 2, rounded: 30},
  {_id: 3, rounded: 30}
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$round` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  try {
    await client.connect();
    const db = client.db('test');
    const collection = db.collection('measurements');
    const result = await collection.aggregate([
      { $project: { rounded: { $round: ["$value", 1] } } }
    ]).toArray();
    console.log(result);
  } finally {
    await client.close();
  }
}
example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    try:
        db = client['test']
        collection = db['measurements']
        result = list(collection.aggregate([
```

```
            {'$project': {'rounded': {'$round': ['$value', 1]}}}
        ]))
        print(result)
    finally:
        client.close()

example()
```

$rtrim

Neu ab Version 4.0.

Wird von Elastic Cluster nicht unterstützt.

Der `$rtrim` Operator in Amazon DocumentDB wird verwendet, um nachfolgende Zeichen aus einer Zeichenfolge zu entfernen. Standardmäßig entfernt er abschließende Leerzeichen, aber Sie können auch eine Reihe von Zeichen angeben, die entfernt werden sollen, indem Sie das Argument chars übergeben.

Parameter

- `input`: Die Eingabezeichenfolge, aus der nachfolgende Leerzeichen entfernt werden sollen.
- `chars`: (optional) Um bestimmte Zeichen zu entfernen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$rtrim` Operator verwendet wird, um nachfolgende angegebene Zeichen (“ *“) aus einer Zeichenfolge zu entfernen.

Erstellen Sie Beispieldokumente

```
db.collection.insert([
  { "name": "John Doe*  ", "age": 30 },
  { "name": "Jane Smith  ", "age": 25 },
  { "name": "Bob Johnson", "age": 35 }
]);
```

Beispiel für eine Abfrage

```
db.collection.aggregate([
```

```
  {
    $project: {
      _id: 0,
      name: { $rtrim: { input: "$name", chars: " *" } },
      age: 1
    }
  }
]);
```

Ausgabe

```
[
  { age: 30, name: 'John Doe' },
  { age: 25, name: 'Jane Smith' },
  { age: 35, name: 'Bob Johnson' }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$rtrim` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient, Bson } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  try {

    const db = client.db('test');
    const collection = db.collection('collection');

    const pipeline = [
      {
        $project: {
          _id: 0,
          name: { $rtrim: { input: "$name", chars: " *" } },
          age: 1
        }
```

```
      }
    ];

    const results = await collection.aggregate(pipeline).toArray();
    console.dir(results, { depth: null });

  } catch (err) {
    console.error('Error occurred:', err);
  } finally {
    await client.close();
  }
}

example().catch(console.error);
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    try:

        db = client.test
        collection = db.collection

        pipeline = [
            {
                "$project": {
                    "_id": 0,
                    "name": { "$rtrim": { "input": "$name", "chars": " *" } },
                    "age": 1
                }
            }
        ]

        results = collection.aggregate(pipeline)

        for doc in results:
            print(doc)
```

```
    except Exception as e:
        print(f"An error occurred: {e}")

    finally:
        client.close()

example()
```

$sample

Die `$sample` Aggregationsphase in Amazon DocumentDB wird verwendet, um nach dem Zufallsprinzip eine bestimmte Anzahl von Dokumenten aus einer Sammlung auszuwählen. Dies ist nützlich für Aufgaben wie Datenanalyse, Tests und Generierung von Stichproben für die weitere Verarbeitung.

Parameter

- `size`: Die Anzahl der Dokumente, die nach dem Zufallsprinzip ausgewählt werden sollen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$sample` Stagingbereich verwendet wird, um nach dem Zufallsprinzip zwei Dokumente aus der `temp` Sammlung auszuwählen.

Erstellen Sie Beispieldokumente

```
db.temp.insertMany([
  { "_id": 1, "temperature": 97.1, "humidity": 0.60, "timestamp":
 ISODate("2019-03-21T21:17:22.425Z") },
  { "_id": 2, "temperature": 98.2, "humidity": 0.59, "timestamp":
 ISODate("2019-03-21T21:17:22.425Z") },
  { "_id": 3, "temperature": 96.8, "humidity": 0.61, "timestamp":
 ISODate("2019-03-21T21:17:22.425Z") },
  { "_id": 4, "temperature": 97.9, "humidity": 0.61, "timestamp":
 ISODate("2019-03-21T21:17:22.425Z") },
  { "_id": 5, "temperature": 97.5, "humidity": 0.60, "timestamp":
 ISODate("2019-03-21T21:17:22.425Z") },
  { "_id": 6, "temperature": 98.0, "humidity": 0.59, "timestamp":
 ISODate("2019-03-21T21:17:22.425Z") },
```

```
  { "_id": 7, "temperature": 97.2, "humidity": 0.60, "timestamp":
 ISODate("2019-03-21T21:17:22.425Z") },
  { "_id": 8, "temperature": 98.1, "humidity": 0.59, "timestamp":
 ISODate("2019-03-21T21:17:22.425Z") },
  { "_id": 9, "temperature": 96.9, "humidity": 0.62, "timestamp":
 ISODate("2019-03-21T21:17:22.425Z") },
  { "_id": 10, "temperature": 97.7, "humidity": 0.60, "timestamp":
 ISODate("2019-03-21T21:17:22.425Z") }
]);
```

Beispiel für eine Abfrage

```
db.temp.aggregate([
   { $sample: { size: 2 } }
])
```

Ausgabe

```
{ "_id" : 4, "temperature" : 97.9, "humidity" : 0.61, "timestamp" :
 ISODate("2019-03-21T21:17:22.425Z") }
{ "_id" : 9, "temperature" : 96.9, "humidity" : 0.62, "timestamp" :
 ISODate("2019-03-21T21:17:22.425Z") }
```

Wie die Ergebnisse zeigen, wurden 2 der 10 Dokumente nach dem Zufallsprinzip ausgewählt. Sie können diese Dokumente nun verwenden, um einen Durchschnitt zu ermitteln oder min/max Berechnungen durchzuführen.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$sample` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function sampleDocuments() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
```

```
  const collection = db.collection('temp');

  const result = await collection.aggregate([
    { $sample: { size: 2 } }
  ]).toArray();

  console.log(result);
  await client.close();
}

sampleDocuments();
```

Python

```
from pymongo import MongoClient

def sample_documents():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['temp']

    result = list(collection.aggregate([
        { '$sample': { 'size': 2 } }
    ]))

    print(result)
    client.close()

sample_documents()
```

$suche

Neu ab Version 5.0.

Der `$search` Operator in Amazon DocumentDB wird verwendet, um Textsuchfunktionen bereitzustellen.

Parameter

Keine

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$search` Operator verwendet wird, um eine Textsuchabfrage durchzuführen.

Erstellen Sie Beispieldokumente

```
db.textcollection.createIndex({"description": "text"});

db.textcollection.insertMany([
  { _id: 1, name: "John Doe", description: "This is a sample document about John
 Doe." },
  { _id: 2, name: "Jane Smith", description: "This is a sample document about Jane
 Smith." },
  { _id: 3, name: "Bob Johnson", description: "This is a sample document about Bob
 Johnson." },
  { _id: 4, name: "Jon Jeffries", description: "This is a sample document about Jon
 Jeffries." }
]);
```

Beispiel für eine Abfrage

```
db.textcollection.find(
  { $text: { $search: "John" } }
);
```

Ausgabe

```
[
  {
    _id: 1,
    name: 'John Doe',
    description: 'This is a sample document about John Doe.'
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$search` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function findWithText() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('textcollection');

  const result = await collection.find(
    { $text: { $search: "John" } }
  ).sort({ score: { $meta: "textScore" } }).toArray();

  console.log(result);
  client.close();
}

findWithText();
```

Python

```
from pymongo import MongoClient

def find_with_text():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['textcollection']

    result = list(collection.find(
        { '$text': { '$search': 'John' } }
    ))

    print(result)
    client.close()

find_with_text()
```

$second

Der `$second` Operator in Amazon DocumentDB extrahiert die Sekundenkomponente aus einem Datums- oder Zeitstempel. Es wird verwendet, um den Sekundenwert aus einem Datums- oder Zeitstempelfeld abzurufen.

Parameter

- `expression`: Das Datums- oder Zeitstempelfeld, aus dem der Sekundenwert extrahiert werden soll. Dieser Ausdruck kann ein Feldpfad oder ein beliebiger gültiger Ausdruck sein, der in ein Datum oder einen Zeitstempel aufgelöst wird.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$second` Operator verwendet wird, um die Sekundenkomponente aus einem Datumsfeld zu extrahieren.

Erstellen Sie Beispieldokumente

```
db.users.insertMany([
  { name: "John", dob: new Date("1990-05-15T12:30:45Z") },
  { name: "Jane", dob: new Date("1985-09-20T23:59:59Z") },
  { name: "Bob", dob: new Date("2000-01-01T00:00:00Z") }
]);
```

Beispiel für eine Abfrage

```
db.users.aggregate([{ $project: { name: 1, dobSeconds: { $second: "$dob" } } }])
```

Ausgabe

```
[
  { "_id" : ObjectId("6089a9c306a829d1f8b456a1"), "name" : "John", "dobSeconds" : 45 },
  { "_id" : ObjectId("6089a9c306a829d1f8b456a2"), "name" : "Jane", "dobSeconds" : 59 },
  { "_id" : ObjectId("6089a9c306a829d1f8b456a3"), "name" : "Bob", "dobSeconds" : 0 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$second` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const users = db.collection('users');

  const result = await users.aggregate([{ $project: { name: 1, dobSeconds:
 { $second: '$dob' } } }]).toArray();
  console.log(result);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    users = db['users']

    result = list(users.aggregate([{'$project': {'name': 1, 'dobSeconds':
 {'$second': '$dob'}}}]))
    print(result)

    client.close()

example ()
```

$set

Neu ab Version 8.0

Wird von Elastic Cluster nicht unterstützt.

Die `$set` Aggregationsphase in Amazon DocumentDB ermöglicht es Ihnen, während einer Aggregationspipeline neue Felder hinzuzufügen oder vorhandene Feldwerte in der Dokumentation zu aktualisieren.

Parameter

- `expression`: Der auszuwertende Ausdruck. Dies kann ein beliebiger gültiger Aggregationsausdruck sein, einschließlich Feldverweise und arithmetischer Operationen.

Beispiel (MongoDB Shell)

Das folgende Beispiel veranschaulicht die Verwendung der `$set` Aggregationsstufe zur Berechnung von Gesamtwerten, indem das Feld mit dem `quantity` Feld multipliziert wird. `price`

Erstellen Sie Beispieldokumente

```
db.inventory.insertMany([
  { item: "pencil", quantity: 100, price: 0.24},
  { item: "pen", quantity: 204, price: 1.78 }
]);
```

Beispiel für eine Aggregation

```
db.inventory.aggregate([
  {
    $set: {
      total: { $multiply: ["$quantity", "$price"] }
    }
  }
])
```

Ausgabe

```
[
  {
    _id: ObjectId('69248951d66dcae121d2950d'),
    item: 'pencil',
    quantity: 100,
    price: 0.24,
    total: 24
```

```
  },
  {
    _id: ObjectId('69248951d66dcae121d2950e'),
    item: 'pen',
    quantity: 204,
    price: 1.78,
    total: 363.12
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$set` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const inventory = db.collection('inventory');

  const result = await inventory.aggregate([
      {
        $set: {
          total: { $multiply: ["$quantity", "$price"] }
        }
      }
  ]).toArray();

  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient
```

```
def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    inventory = db['inventory']

    result = list(inventory.aggregate([
      {
        "$set": {
          "total": { "$multiply": ["$quantity", "$price"] }
        }
      }
    ]))

    print(result)
    client.close()

example()
```

$setDifference

Neu ab Version 4.0.

Der `$setDifference` Operator in Amazon DocumentDB wird verwendet, um zwei Sätze zu vergleichen und die Elemente zurückzugeben, die sich in der ersten Gruppe, aber nicht in der zweiten Gruppe befinden. Dieser Operator ist nützlich, um die eindeutigen Elemente zwischen zwei Sätzen zu finden.

Parameter

- `firstSet`: Der erste Satz, der verglichen werden soll.
- `secondSet`: Der zweite zu vergleichende Satz.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$setDifference` Operator verwendet wird, um die eindeutigen Elemente zwischen zwei Sätzen zu finden.

Erstellen Sie Beispieldokumente

```
db.collection.insertMany([
  { _id: 1, fruits: ["apple", "banana", "cherry", "date"] },
  { _id: 2, fruits: ["banana", "cherry", "date", "elderberry"] }
]);
```

Beispiel für eine Abfrage

```
db.collection.aggregate([
  {
    $project: {
      uniqueFruits: { $setDifference: ["$fruits", ["banana", "cherry", "date"]] }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "uniqueFruits": ["apple"] },
  { "_id": 2, "uniqueFruits": ["elderberry"] }
]
```

Die Abfrage führt die folgenden Schritte aus:

1. Sie verwendet die `$project` Phase, um `uniqueFruits` für jedes Dokument ein neues Feld zu erstellen.

2. Der `$setDifference` Operator vergleicht das `fruits` Array mit dem Array `["banana", "cherry", "date"]` und gibt die eindeutigen Elemente im `fruits` Array zurück.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$setDifference` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

Hier ist ein Beispiel für die Verwendung des `$setDifference` Operators in einer Node.js Anwendung:

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('mycollection');

  // Insert sample documents
  await collection.insertMany([
    { _id: 1, fruits: ["apple", "banana", "cherry", "date"] },
    { _id: 2, fruits: ["banana", "cherry", "date", "elderberry"] }
  ]);

 // Query using $setDifference
  const result = await collection.aggregate([
    {
      $project: {
        uniqueFruits: { $setDifference: ["$fruits", ["banana", "cherry", "date"]] }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

main();
```

Python

Hier ist ein Beispiel für die Verwendung des `$setDifference` Operators in einer Python-Anwendung:

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['mycollection']
```

```
    # Insert sample documents
    collection.insert_many([
        {'_id': 1, 'fruits': ["apple", "banana", "cherry", "date"]},
        {'_id': 2, 'fruits': ["banana", "cherry", "date", "elderberry"]}
    ])

    # Query using $setDifference
    result = list(collection.aggregate([
        {
            '$project': {
                'uniqueFruits': {'$setDifference': ['$fruits', ["banana", "cherry",
 "date"]]}
            }
        }
    ]))

    print(result)
    client.close()

if __name__ == '__main__':
    main()
```

$setEquals

Der `$setEquals` Operator in Amazon DocumentDB wird verwendet, um festzustellen, ob zwei Sätze gleich sind. Er vergleicht zwei Arrays und gibt zurück, `true` ob sie dieselben unterschiedlichen Elemente enthalten, unabhängig von ihrer Reihenfolge.

Parameter

- `expression1`: Das erste Array, das verglichen werden soll.
- `expression2`: Das zweite Array, das verglichen werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$setEquals` Operator verwendet wird, um zwei Wertesätze zu vergleichen.

Erstellen Sie Beispieldokumente

```
db.collection.insertMany([
  { _id: 1, fruits: ["apple", "banana", "cherry"] },
  { _id: 2, fruits: ["banana", "apple", "cherry"] },
  { _id: 3, fruits: ["apple", "banana", "orange"] }
])
```

Beispiel für eine Abfrage

```
db.collection.find({
  $expr: {
    $setEquals: ["$fruits", ["apple", "banana", "cherry"]]
  }
})
```

Ausgabe

```
{ "_id" : 1, "fruits" : [ "apple", "banana", "cherry" ] }
{ "_id" : 2, "fruits" : [ "banana", "apple", "cherry" ] }
```

Die Abfrage verwendet den `$setEquals` Operator, um das `fruits` Feld jedes Dokuments mit dem Array zu vergleichen`["apple", "banana", "cherry"]`. Die Dokumente, bei denen das `fruits` Feld dem Vergleichsarray entspricht, werden zurückgegeben.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$setEquals` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('mycollection');

  // Insert sample documents
```

```
  await collection.insertMany([
    { _id: 1, fruits: ["apple", "banana", "cherry"] },
    { _id: 2, fruits: ["banana", "apple", "cherry"] },
    { _id: 3, fruits: ["apple", "banana", "orange"] }
  ]);

  // Query using $setEquals
  const result = await collection.find({
    $expr: {
      $setEquals: ["$fruits", ["apple", "banana", "cherry"]]
    }
  }).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['mycollection']

    # Insert sample documents
    collection.insert_many([
        {"_id": 1, "fruits": ["apple", "banana", "cherry"]},
        {"_id": 2, "fruits": ["banana", "apple", "cherry"]},
        {"_id": 3, "fruits": ["apple", "banana", "orange"]}
    ])

    # Query using $setEquals
    result = list(collection.find({
        "$expr": {
            "$setEquals": ["$fruits", ["apple", "banana", "cherry"]]
        }
    }))
```

```
    print(result)
    client.close()

example()
```

$setIntersection

Der `$setIntersection` Operator in Amazon DocumentDB wird verwendet, um die gemeinsamen Elemente zwischen zwei oder mehr Arrays zurückzugeben. Dieser Operator ist besonders nützlich, wenn Sie mit Datensätzen arbeiten. Er ermöglicht es Ihnen, die Schnittmenge mehrerer Datensätze zu ermitteln.

Parameter

- `array1`: Das erste Array, das sich überschneidet.
- `array2`: Das zweite Array, das sich überschneiden soll.
- `arrayN`: (optional) Zusätzliche Arrays, die sich überschneiden sollen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$setIntersection` Operator verwendet wird, um die gemeinsamen Elemente zwischen zwei Arrays zu finden.

Erstellen Sie Beispieldokumente

```
db.collection.insertMany([
  { _id: 1, colors: ["red", "blue", "green"] },
  { _id: 2, colors: ["blue", "yellow", "orange"] },
  { _id: 3, colors: ["red", "green", "purple"] }
])
```

Beispiel für eine Abfrage

```
db.collection.aggregate([
  { $project: {
      _id: 1,
      commonColors: { $setIntersection: ["$colors", ["red", "blue", "green"]] }
```

```
    }
  }
])
```

Ausgabe

```
[
  { "_id": 1, "commonColors": ["red", "blue", "green"] },
  { "_id": 2, "commonColors": ["blue"] },
  { "_id": 3, "commonColors": ["red", "green"] }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$setIntersection` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('mycollection');

  const result = await collection.aggregate([
    {
      $project: {
        _id: 1,
        commonColors: { $setIntersection: ["$colors", ["red", "blue", "green"]] }
      }
    }
  ]).toArray();

  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['mycollection']

    result = list(collection.aggregate([
        {
            '$project': {
                '_id': 1,
                'commonColors': { '$setIntersection': ["$colors", ["red", "blue",
 "green"]] }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$ gesetzt IsSubset

Der `$setIsSubset` Operator in Amazon DocumentDB wird verwendet, um festzustellen, ob eine Gruppe von Werten eine Teilmenge einer anderen Menge ist. Er ist nützlich, um mengenbasierte Vergleiche und Operationen mit Array-Feldern durchzuführen.

Parameter

- `field`: Das Feld, auf das der `$setIsSubset` Operator angewendet werden soll.
- `set`: Das Set, mit dem das Feld verglichen werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$setIsSubset` Operator verwendet wird, um zu überprüfen, ob das `tags` Feld eine Teilmenge der angegebenen Menge ist.

Erstellen Sie Beispieldokumente

```
db.products.insertMany([
  { _id: 1, name: "Product A", tags: ["tag1", "tag2", "tag3"] },
  { _id: 2, name: "Product B", tags: ["tag1", "tag2"] },
  { _id: 3, name: "Product C", tags: ["tag2", "tag3"] }
]);
```

Beispiel für eine Abfrage

```
db.products.find({
  $expr: { $setIsSubset: [["tag1", "tag2"], "$tags"] }
})
```

*Hinweis: * `$setIsSubset` ist ein Aggregationsoperator und kann nicht direkt in find () -Abfragen verwendet werden. In diesem Beispiel `$expr` wird mit verwendet, `find()` um die Lücke zwischen Abfrageoperatoren und Aggregationsausdrücken zu schließen.

Ausgabe

```
[
  { "_id" : 1, "name" : "Product A", "tags" : [ "tag1", "tag2", "tag3" ] },
  { "_id" : 2, "name" : "Product B", "tags" : [ "tag1", "tag2" ] }
]
```

Die Abfrage gibt die Dokumente zurück, bei denen das `tags` Feld eine Teilmenge der Menge ist. `["tag1", "tag2"]`

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$setIsSubset` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
```

```
  const collection = db.collection('products');

  const result = await collection.find({
    $expr: { $setIsSubset: [["tag1", "tag2"], "$tags"] }
  }).toArray();

  console.log(result);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['products']

    result = list(collection.find({
        '$expr': {'$setIsSubset': [['tag1', 'tag2'], '$tags']}
    }))

    print(result)

    client.close()

example()
```

$setUnion

Der `$setUnion` Aggregationsoperator in Amazon DocumentDB wird verwendet, um zwei oder mehr Wertesätze zu kombinieren und eine Menge zurückzugeben, die alle eindeutigen Elemente aus den Eingabesätzen enthält. Dieser Operator ist nützlich, wenn Sie mengenbasierte Operationen für Array-Felder in Ihren Dokumenten ausführen müssen.

Parameter

- `expression1`: Ein Ausdruck, der in ein Array aufgelöst wird.
- `expression2`: Ein Ausdruck, der in ein Array aufgelöst wird.
- `expressionN`: Zusätzliche Ausdrücke, die in Arrays aufgelöst werden (optional).

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$setUnion` Operator verwendet wird, um die eindeutigen Elemente aus zwei Array-Feldern in einer Sammlung zu kombinieren.

Erstellen Sie Beispieldokumente

```
db.users.insertMany([
  { _id: 1, name: "Alice", hobbies: ["reading", "swimming"], skills: ["coding",
 "writing"] },
  { _id: 2, name: "Bob", hobbies: ["cooking", "gardening"], skills: ["coding",
 "photography"] },
  { _id: 3, name: "Charlie", hobbies: ["reading", "painting"], skills: ["gardening",
 "music"] }
]);
```

Beispiel für eine Abfrage

```
db.users.aggregate([
  {
    $project: {
      name: 1,
      allInterests: { $setUnion: ["$hobbies", "$skills"] }
    }
  }
]);
```

Ausgabe

```
[
  { "_id" : 1, "name" : "Alice", "allInterests" : [ "coding", "reading", "swimming",
 "writing" ] },
  { "_id" : 2, "name" : "Bob", "allInterests" : [ "coding", "cooking", "gardening",
 "photography" ] },
  { "_id" : 3, "name" : "Charlie", "allInterests" : [ "gardening", "music", "painting",
 "reading" ] }
```

```
]
```

In diesem Beispiel wird der `$setUnion` Operator verwendet, um die eindeutigen Elemente aus den Feldern `hobbies` und `skills` Array für jedes Benutzerdokument zu kombinieren. Das resultierende `allInterests` Feld enthält die Zusammenfassung aller einzigartigen Hobbys und Fähigkeiten für jeden Benutzer.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$setUnion` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const users = db.collection('users');

  const result = await users.aggregate([
    {
      $project: {
        _id: 1,
        name: 1,
        allInterests: { $setUnion: ["$hobbies", "$skills"] }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient
```

```
def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    users = db['users']

    result = list(users.aggregate([
        {
            '$project': {
                '_id': 1,
                'name': 1,
                'allInterests': { '$setUnion': ["$hobbies", "$skills"] }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$ setzen WindowFields

Neu ab Version 8.0.2.

Die `$setWindowFields` Phase in Amazon DocumentDB führt Operationen für einen bestimmten Bereich von Dokumenten (ein Fenster) in einer Partition durch und gibt die Ergebnisse auf der Grundlage des ausgewählten Fensteroperators zurück. Die Phase gruppiert die Eingabedokumente in Partitionen, sortiert sie optional und berechnet für jedes Dokument ein oder mehrere Ausgabefelder über ein Fenster mit umgebenden Dokumenten.

Syntax

```
{
  $setWindowFields: {
    partitionBy: <expression>,
    sortBy: { <sortField1>: <sortOrder>, ... },
    output: {
      <outputField1>: {
        <windowOperator>: <expression>,
```

```
        window: {
          documents: [ <lowerBound>, <upperBound> ],
          range: [ <lowerBound>, <upperBound> ],
          unit: <time unit>
        }
      },
      ...
    }
  }
}
```

Parameter

- `partitionBy`: Optional. Ein einzelner Ausdruck, der verwendet wird, um die Dokumente in Partitionen zu gruppieren. Dies kann ein Feldpfad (z. B.`"$state"`) oder ein anderer Ausdruck sein. Wenn es weggelassen wird, gehören alle Eingabedokumente zu einer einzigen Partition.
- `sortBy`: Optional. Ein Dokument, das das Feld oder die Felder angibt, nach denen die Dokumente in jeder Partition sortiert werden sollen, und zwar in der Form `{ <field>: 1 }` für aufsteigende oder `{ <field>: -1 }` für absteigende Reihenfolge. A `sortBy` ist erforderlich, wenn Sie dokument- oder bereichsbasierte Fenstergrenzen verwenden und wenn Sie einen der Operatoren für die Rangfolge verwenden.
- `output`: Erforderlich. Ein Dokument, das die Felder angibt, die an jedes Dokument angehängt werden sollen. Sie können mehrere Ausgabefelder angeben, aber jedes Ausgabefeld benötigt genau einen Fensteroperator, sein Argument und ein optionales `window` Dokument, das die Fenstergrenzen definiert:`{ <field>: { <windowOperator>: <expression>, window: { ... } } }`.

Fenstergrenzen

In jedem `output` Feld können Sie optional die Fenstergrenzen mit einem `window` Dokument definieren. Ein Fenster kann anhand der Dokumentposition oder anhand eines Wertebereichs definiert werden.

- `documents`: Gibt das Fenster als eine Anzahl von Dokumenten vor und nach dem aktuellen Dokument im Formular an`[ <lowerBound>, <upperBound> ]`. Jede Grenze ist ein Ganzzahl-Offset (z. B. `-1` oder2) oder eines der Schlüsselwörter `"unbounded"` und`"current"`. `"unbounded"`erweitert das Fenster bis zum Anfang der Partition, wenn es als Untergrenze verwendet wird, oder bis zum Ende der Partition, wenn es als Obergrenze verwendet wird.

`"current"`bedeutet das aktuelle Dokument. Ein `documents` Fenster benötigt einen, `sortBy` es sei denn, beide Grenzen sind `"unbounded"` gleich.

- `range`: Gibt das Fenster als einen Bereich von `sortBy` Feldwerten im Verhältnis zum aktuellen Dokument im Formular `[ <lowerBound>, <upperBound> ]` an. Jede Grenze ist eine Zahl, die dem `sortBy` Wert des aktuellen Dokuments oder dem `"unbounded"` Schlüsselwort hinzugefügt wird. Ein `range` Fenster benötigt genau ein aufsteigendes `sortBy` Feld und akzeptiert das `"current"` Schlüsselwort nicht.
- `unit`: Optional. Wird verwendet`range`, um die numerischen Fenstergrenzen als Zeitspanne zu interpretieren, in der das `sortBy` Feld Datumswerte enthält. Amazon DocumentDB unterstützt die folgenden Einheiten: `"year""quarter","month","week",,"day", "hour" "minute""second"`, und. `"millisecond"` Wenn `unit` angegeben, müssen die `range` Grenzen ganze Zahlen sein.

Wenn das `window` Dokument weggelassen wird, wird das Standardfenster verwendet`documents: ["unbounded", "unbounded"]`, das sich über die gesamte Partition erstreckt.

> **Note**
>
> Wenn Sie ein `range` Fenster verwenden, das Datumswerten vorzieht, empfehlen wir eine feste Dauer `unit` wie `"day"` (oder kleiner) und nicht, oder `"month" "quarter""year"`, deren kalenderabhängige Länge zu mehrdeutigen Fenstergrenzen führen kann.

Unterstützte Fensteroperatoren

Amazon DocumentDB unterstützt die folgenden 26 Fensteroperatoren im `output` Bereich der `$setWindowFields` Bühne:

- `$sum`
- `$avg`
- `$count`
- `$push`
- `$addToSet`
- `$stdDevPop`
- `$stdDevSamp`

- `$covariancePop`
- `$covarianceSamp`
- `$first`
- `$last`
- `$firstN`
- `$lastN`
- `$top`
- `$bottom`
- `$topN`
- `$bottomN`
- `$min`
- `$minN`
- `$max`
- `$maxN`
- `$median`
- `$percentile`
- `$rank`
- `$denseRank`
- `$documentNumber`

> **Note**
>
> Der `$rand` Operator wird innerhalb eines Fensteroperators in der `$setWindowFields` Bühne nicht unterstützt.

Leere Fenster

Wenn ein Fenster keine Dokumente enthält (z. B. ein `range` Fenster ohne Werte im Bereich), hängt das Ergebnis vom Operator ab: `$count` und gibt `$sum` zurück`0`;`$push`,,,,,,, `$addToSet` `$firstN` `$lastN` `$minN` `$maxN``$topN`, und gibt ein leeres Array `$percentile` zurück; gibt ein Array von

`null` Werten zurück (einen pro angefordertem Perzentil); alle anderen Operatoren `$bottomN` kehren zurück. `null`

Beispiel (MongoDB Shell)

Das folgende Beispiel partitioniert die Dokumente nach`region`, sortiert jede Partition nach `day` und berechnet eine laufende Summe `revenue` vom Anfang der Partition bis zum aktuellen Dokument.

Erstellen Sie Beispieldokumente

```
db.sales.insertMany([
  { _id: 1, region: "east", day: 1, revenue: 100 },
  { _id: 2, region: "east", day: 2, revenue: 150 },
  { _id: 3, region: "east", day: 3, revenue: 120 },
  { _id: 4, region: "west", day: 1, revenue: 200 },
  { _id: 5, region: "west", day: 2, revenue: 180 }
]);
```

Beispiel für eine Abfrage

```
db.sales.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$region",
      sortBy: { day: 1 },
      output: {
        runningRevenue: {
          $sum: "$revenue",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "region": "east", "day": 1, "revenue": 100, "runningRevenue": 100 },
  { "_id": 2, "region": "east", "day": 2, "revenue": 150, "runningRevenue": 250 },
  { "_id": 3, "region": "east", "day": 3, "revenue": 120, "runningRevenue": 370 },
  { "_id": 4, "region": "west", "day": 1, "revenue": 200, "runningRevenue": 200 },
  { "_id": 5, "region": "west", "day": 2, "revenue": 180, "runningRevenue": 380 }
```

```
]
```

Jedes Dokument wird um die kumulative Summe `revenue` innerhalb seiner Partition bis einschließlich des aktuellen Dokuments erweitert. `runningRevenue`

Codebeispiele

Um ein Codebeispiel für die Verwendung der `$setWindowFields` Bühne anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  try {
    await client.connect();
    const db = client.db('test');
    const sales = db.collection('sales');

    await sales.insertMany([
      { _id: 1, region: "east", day: 1, revenue: 100 },
      { _id: 2, region: "east", day: 2, revenue: 150 },
      { _id: 3, region: "east", day: 3, revenue: 120 },
      { _id: 4, region: "west", day: 1, revenue: 200 },
      { _id: 5, region: "west", day: 2, revenue: 180 }
    ]);

    const result = await sales.aggregate([
      {
        $setWindowFields: {
          partitionBy: "$region",
          sortBy: { day: 1 },
          output: {
            runningRevenue: {
              $sum: "$revenue",
              window: { documents: ["unbounded", "current"] }
            }
          }
```

```
        }
      }
    ]).toArray();

    console.log(result);
  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    try:
        db = client['test']
        sales = db['sales']

        sales.insert_many([
            { '_id': 1, 'region': 'east', 'day': 1, 'revenue': 100 },
            { '_id': 2, 'region': 'east', 'day': 2, 'revenue': 150 },
            { '_id': 3, 'region': 'east', 'day': 3, 'revenue': 120 },
            { '_id': 4, 'region': 'west', 'day': 1, 'revenue': 200 },
            { '_id': 5, 'region': 'west', 'day': 2, 'revenue': 180 }
        ])

        result = list(sales.aggregate([
            {
                '$setWindowFields': {
                    'partitionBy': '$region',
                    'sortBy': { 'day': 1 },
                    'output': {
                        'runningRevenue': {
                            '$sum': '$revenue',
                            'window': { 'documents': ['unbounded', 'current'] }
                        }
```

```
                    }
                }
            }
        ]))

        print(result)
    finally:
        client.close()

example()
```

$ sigmoid

Neu ab Version 8.0.1.

Der `$sigmoid` Operator in Amazon DocumentDB wendet die logistische Sigmoidfunktion,`1 / (1 + e^(-x))`, auf eine numerische Eingabe an. Die Funktion ordnet jede reelle Zahl einem Wert zwischen 0 und 1 zu und erzeugt so eine Kurve. S-shaped Dies ist nützlich, wenn Sie Werte im Bereich von 0—1 normalisieren müssen, z. B. um Rohwerte in Wahrscheinlichkeiten oder Relevanzwerte umzuwandeln.

Parameter

- `expression`: Ein Ausdruck, der in einen numerischen Wert aufgelöst wird.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$sigmoid` Operator verwendet wird, um Logit-Werte in Wahrscheinlichkeiten umzuwandeln.

Erstellen Sie Beispieldokumente

```
db.predictions.insertMany([
  {_id: 1, logit: -2},
  {_id: 2, logit: 0},
  {_id: 3, logit: 2}
]);
```

Beispiel für eine Abfrage

```
db.predictions.aggregate([
```

```
  { $project: { probability: { $sigmoid: "$logit" } } }
]);
```

Ausgabe

```
[
  {_id: 1, probability: 0.11920292202211755},
  {_id: 2, probability: 0.5},
  {_id: 3, probability: 0.8807970779778823}
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$sigmoid` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  try {
    await client.connect();
    const db = client.db('test');
    const collection = db.collection('predictions');
    const result = await collection.aggregate([
      { $project: { probability: { $sigmoid: "$logit" } } }
    ]).toArray();
    console.log(result);
  } finally {
    await client.close();
  }
}
example();
```

Python

```
from pymongo import MongoClient
```

```
def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    try:
        db = client['test']
        collection = db['predictions']
        result = list(collection.aggregate([
            {'$project': {'probability': {'$sigmoid': '$logit'}}}
        ]))
        print(result)
    finally:
        client.close()

example()
```

$sin

Neu ab Version 8.0.1.

Der `$sin` Operator in Amazon DocumentDB gibt den Sinus eines Werts zurück, der im Bogenmaß gemessen wird. Verwenden Sie ihn in der Aggregationspipeline, um trigonometrische Berechnungen für numerische Felder durchzuführen.

Parameter

- `expression`: Ein Ausdruck, der in eine Zahl im Bogenmaß aufgelöst wird.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$sin` Operator verwendet wird, um den Sinus von Winkelwerten im Bogenmaß zu berechnen.

Musterdokumente erstellen

```
db.angles.insertMany([
  { "_id": 1, "angle": 0 },
  { "_id": 2, "angle": 0.5235987755982988 },
  { "_id": 3, "angle": 1.5707963267948966 }
]);
```

Beispiel für eine Abfrage

```
db.angles.aggregate([
  { $project: {
    "sine": { $sin: "$angle" }
  }}
]);
```

Ausgabe

```
[
  { "_id": 1, "sine": 0 },
  { "_id": 2, "sine": 0.49999999999999994 },
  { "_id": 3, "sine": 1 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$sin` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('angles');

  const result = await collection.aggregate([
    { $project: {
      "sine": { $sin: "$angle" }
    }}
  ]).toArray();

  console.log(result);
  await client.close();
}
```

```
main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['angles']

    result = list(collection.aggregate([
        { '$project': {
            'sine': { '$sin': '$angle' }
        }}
    ]))

    print(result)
    client.close()

if __name__ == "__main__":
    main()
```

$sinh

Neu ab Version 8.0.1.

Der `$sinh` Operator in Amazon DocumentDB gibt den hyperbolischen Sinus eines Werts zurück. Verwenden Sie ihn in der Aggregationspipeline, um hyperbolische trigonometrische Berechnungen für numerische Felder durchzuführen.

Parameter

- `expression`: Ein Ausdruck, der in eine Zahl aufgelöst wird.

Der Rückgabetyp ist `double` standardmäßig. Wenn die Eingabe eine 128-Bit-Dezimalzahl ist, ist die Ausgabe ebenfalls eine 128-Bit-Dezimalzahl.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$sinh` Operator verwendet wird, um den hyperbolischen Sinus numerischer Werte zu berechnen.

Erstellen Sie Beispieldokumente

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 1 },
  { "_id": 3, "value": -1 }
]);
```

Beispiel für eine Abfrage

```
db.values.aggregate([
  { $project: {
    "result": { $sinh: "$value" }
  }}
]);
```

Ausgabe

```
[
  { "_id": 1, "result": 0 },
  { "_id": 2, "result": 1.1752011936438014 },
  { "_id": 3, "result": -1.1752011936438014 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$sinh` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
```

```
  const db = client.db('test');
  const collection = db.collection('values');

  const result = await collection.aggregate([
    { $project: {
      "result": { $sinh: "$value" }
    }}
  ]).toArray();

  console.log(result);
  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['values']

    result = list(collection.aggregate([
        { '$project': {
            'result': { '$sinh': '$value' }
        }}
    ]))

    print(result)
    client.close()

if __name__ == "__main__":
    main()
```

$skip

In Amazon DocumentDB wird der `$skip` Operator verwendet, um den Startpunkt der Abfrageergebnisse zu verschieben, sodass Sie eine bestimmte Teilmenge der passenden

Dokumente abrufen können. Dies ist besonders nützlich in Paginierungsszenarien, in denen Sie nachfolgende Ergebnisseiten abrufen möchten.

Parameter

- `skip`: Die Anzahl der Dokumente, die übersprungen werden sollen, bevor die verbleibenden Dokumente zurückgegeben werden.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$skip` Operator verwendet wird, um die zweite Ergebnisseite (Dokumente 11-20) aus einer Sammlung abzurufen.

Erstellen Sie Beispieldokumente

```
db.collection.insert([
  { "name": "Document 1" },
  { "name": "Document 2" },
  { "name": "Document 3" },
  { "name": "Document 4" },
  { "name": "Document 5" },
  { "name": "Document 6" },
  { "name": "Document 7" },
  { "name": "Document 8" },
  { "name": "Document 9" },
  { "name": "Document 10" },
  { "name": "Document 11" },
  { "name": "Document 12" },
  { "name": "Document 13" },
  { "name": "Document 14" },
  { "name": "Document 15" },
  { "name": "Document 16" },
  { "name": "Document 17" },
  { "name": "Document 18" },
  { "name": "Document 19" },
  { "name": "Document 20" }
]);
```

Beispiel für eine Abfrage

```
db.collection.find({}, { "name": 1 })
```

```
            .skip(10)
            .limit(10);
```

Ausgabe

```
[
  { "_id" : ObjectId("..."), "name" : "Document 11" },
  { "_id" : ObjectId("..."), "name" : "Document 12" },
  { "_id" : ObjectId("..."), "name" : "Document 13" },
  { "_id" : ObjectId("..."), "name" : "Document 14" },
  { "_id" : ObjectId("..."), "name" : "Document 15" },
  { "_id" : ObjectId("..."), "name" : "Document 16" },
  { "_id" : ObjectId("..."), "name" : "Document 17" },
  { "_id" : ObjectId("..."), "name" : "Document 18" },
  { "_id" : ObjectId("..."), "name" : "Document 19" },
  { "_id" : ObjectId("..."), "name" : "Document 20" }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$skip` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('collection');

  const results = await collection.find({}, { projection: { name: 1 } })
                                  .skip(10)
                                  .limit(10)
                                  .toArray();

  console.log(results);

  await client.close();
```

```
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.mydatabase
    collection = db.collection

    results = list(collection.find({}, {'name': 1})
                   .skip(10)
                   .limit(10))

    print(results)

    client.close()

if __name__ == '__main__':
    main()
```

$slice

Mit dem `$slice` Aggregationsoperator können Sie eine Teilmenge eines Arrays zurückgeben, indem Sie das Array entweder vom Anfang oder vom Ende des Arrays durchqueren. Dies wird verwendet, um eine begrenzte Anzahl von Elementen aus einem Array-Feld anzuzeigen, z. B. die obersten oder untersten N Elemente.

Parameter

- `array`: Das Array-Feld, das aufgeteilt werden soll.
- n: Eine Ganzzahl, die die Anzahl der zurückzugebenden Elemente angibt. Ein positiver Wert beginnt am Anfang des Arrays, während ein negativer Wert am Ende des Arrays beginnt.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie Sie `$slice` jedem Koch die ersten beiden Lieblingssüßigkeiten zurückgeben können.

Erstellen Sie Beispieldokumente

```
db.sweets.insertMany([
  { "_id" : 1, "name" : "Alvin", "favorites": [ "chocolate", "cake", "toffee",
 "beignets" ] },
  { "_id" : 2, "name" : "Tom", "favorites": [ "donuts", "pudding", "pie" ] },
  { "_id" : 3, "name" : "Jessica", "favorites": [ "fudge", "smores", "pudding",
 "cupcakes" ] },
  { "_id" : 4, "name" : "Rachel", "favorites": [ "ice cream" ] }
]);
```

Beispiel für eine Abfrage

```
db.sweets.aggregate([
  { $project: { _id: 0, name: 1, topTwoFavorites: { $slice: [ "$favorites", 2 ] } } }
]);
```

Ausgabe

```
[
  { name: 'Alvin', topTwoFavorites: [ 'chocolate', 'cake' ] },
  { name: 'Tom', topTwoFavorites: [ 'donuts', 'pudding' ] },
  { name: 'Jessica', topTwoFavorites: [ 'fudge', 'smores' ] },
  { name: 'Rachel', topTwoFavorites: [ 'ice cream' ] }
]
```

In diesem Beispiel wird der `$slice` Operator verwendet, um die ersten beiden Elemente aus dem `favorites` Array für jedes Dokument zu extrahieren.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$slice` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');
```

```
async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('sweets');

  const result = await collection.aggregate([
    { $project: { name: 1, topTwoFavorites: { $slice: ['$favorites', 2] } } }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['sweets']

    result = list(collection.aggregate([
        { '$project': { 'name': 1, 'topTwoFavorites': { '$slice': ['$favorites',
 2] } } }
    ]))

    print(result)
    client.close()

example()
```

$size

Der `$size` Operator wird verwendet, um die Anzahl der Elemente in einem Array-Feld zurückzugeben. Dies kann verwendet werden, um die Anzahl der Elemente in einem Array zu bestimmen, das in einem Dokument gespeichert ist.

Parameter

- `field`: Der Feldpfad, dessen Array-Größe Sie zurückgeben möchten.

Beispiel (MongoDB Shell)

Dieses Beispiel zeigt, wie der `$size` Operator verwendet wird, um die Anzahl der Teams zurückzugeben, denen jeder Benutzer folgt.

Erstellen Sie Beispieldokumente

```
db.profiles.insertMany([
  { _id: 1, name: "John Doe", teams: ["Acme", "Widgets", "Gadgets"] },
  { _id: 2, name: "Jane Smith", teams: ["Acme", "Gadgets"] },
  { _id: 3, name: "Bob Johnson", teams: ["Acme", "Widgets", "Gadgets"] }
]);
```

Beispiel für eine Abfrage

```
db.profiles.aggregate([
  {
    $project: {
      _id: 0,
      name: 1,
      "numberOfTeams": { $size: "$teams" }
    }
  }
])
```

Ausgabe

```
[
  { name: 'John Doe', numberOfTeams: 3 },
  { name: 'Jane Smith', numberOfTeams: 2 },
```

```
  { name: 'Bob Johnson', numberOfTeams: 3 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$size` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const profiles = db.collection('profiles');

  const result = await profiles.aggregate([
    {
      $project: {
        item: 1,
        "numberOfTeams": { $size: "$teams" }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
```

```
    profiles = db.profiles

    result = list(profiles.aggregate([
        {
            '$project': {
                'item': 1,
                'numberOfTeams': { '$size': '$teams' }
            }
        }
    ]))

    print(result)
    client.close()

if __name__ == "__main__":
    main()
```

$sort

In der `$sort` Aggregationsphase werden Dokumente in der Pipeline anhand der angegebenen Feldwerte sortiert. Die Dokumente werden gemäß den angegebenen Sortierkriterien in auf- oder absteigender Reihenfolge angeordnet.

Parameter

- `field`: Der Feldname, nach dem sortiert werden soll.
- `order`: Wird 1 für aufsteigende oder `-1` absteigende Reihenfolge verwendet.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie die `$sort` Bühne verwendet wird, um Produkte nach Preis in absteigender Reihenfolge zu sortieren.

Erstellen Sie Beispieldokumente

```
db.products.insertMany([
  { _id: 1, name: "Laptop", category: "Electronics", price: 1200 },
  { _id: 2, name: "Mouse", category: "Electronics", price: 25 },
  { _id: 3, name: "Desk", category: "Furniture", price: 350 },
  { _id: 4, name: "Chair", category: "Furniture", price: 150 },
  { _id: 5, name: "Monitor", category: "Electronics", price: 400 }
```

```
]);
```

Beispiel für eine Abfrage

```
db.products.aggregate([
  { $sort: { price: -1 } }
]);
```

Ausgabe

```
[
  { _id: 1, name: 'Laptop', category: 'Electronics', price: 1200 },
  { _id: 5, name: 'Monitor', category: 'Electronics', price: 400 },
  { _id: 3, name: 'Desk', category: 'Furniture', price: 350 },
  { _id: 4, name: 'Chair', category: 'Furniture', price: 150 },
  { _id: 2, name: 'Mouse', category: 'Electronics', price: 25 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung der `$sort` Aggregationsstufe anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('products');

  const result = await collection.aggregate([
    { $sort: { price: -1 } }
  ]).toArray();

  console.log(result);
  await client.close();
}
```

```
example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['products']

    result = list(collection.aggregate([
        { '$sort': { 'price': -1 } }
    ]))

    print(result)
    client.close()

example()
```

Inkrementelle Sortierung

Die inkrementelle Sortierung ist eine Optimierung, bei der Amazon DocumentDB einen Index verwendet, der bereits die führenden Felder einer Sortierung mit mehreren Feldern sortiert. Amazon DocumentDB sortiert dann nur die verbleibenden Felder innerhalb jeder Gruppe von Dokumenten, die dieselben führenden Werte aufweisen. Anstatt die gesamte Ergebnismenge im Speicher in einem Block zu sortieren (eine vollständige Sortierung), liest Amazon DocumentDB Dokumente in Indexreihenfolge. Es führt dann viele kleine, unabhängige Sortierungen durch, eine pro Gruppe. Die Ergebnisse sind identisch mit einer vollständigen Sortierung. Nur der Abfrageplan und die Menge an Arbeitsspeicher und Arbeit, die er verwendet, ändern sich.

Vorteile

Die inkrementelle Sortierung bietet die folgenden Vorteile:

- Geringerer Speicherverbrauch und weniger Probleme bei der datenträgerbasierten Sortierung. Amazon DocumentDB bestellt jeweils eine kleine Gruppe und nicht den gesamten Ergebnissatz auf einmal.

- Schnellere Ergebnisse mit**`limit`**. Amazon DocumentDB kann sortierte Dokumente ausgeben, bevor es die gesamte Eingabe liest. Eine Sortierung, gefolgt von einer `limit` (z. B. eine Top-N- oder paginierte Abfrage), kann beendet werden, sobald genügend Dokumente erstellt wurden.

Wenn Amazon DocumentDB eine inkrementelle Sortierung auswählt

Wenn beispielsweise die Sortierung aktiviert `{ customerId: 1, orderDate: -1 }` und nur der Index aktiviert ist, gibt der Index bereits Dokumente zurück`{ customerId: 1 }`, die nach sortiert sind. `customerId` Amazon DocumentDB muss nur jede Gruppe von Dokumenten, die dasselbe Format haben, `customerId` nach `orderDate` sortieren. Das bereits geordnete führende Feld (`customerId`) ist der vorsortierte Schlüssel. Je nachdem, wie viel Sortierung ein Index bereits bietet, wählt Amazon DocumentDB eines von drei Verhaltensweisen:

- Der Index sortiert jeden Sortierschlüssel. Es ist keine Sortierung erforderlich. Amazon DocumentDB liest Dokumente direkt in der angeforderten Reihenfolge.
- Der Index ordnet ein vorangestelltes Präfix der Sortierschlüssel an. Amazon DocumentDB verwendet eine inkrementelle Sortierung und sortiert nur die nachfolgenden Schlüssel innerhalb jeder vorsortierten Gruppe.
- Kein Index sortiert nach einem führenden Sortierschlüssel. Amazon DocumentDB führt eine vollständige Sortierung der gesamten Eingabe durch.

Anforderungen und Einschränkungen

Für die inkrementelle Sortierung ist Amazon DocumentDB 8.0.2 oder höher erforderlich, wobei die Planner-Version 3.0 ausgeführt wird (der Standard-Abfrageplaner ab Amazon DocumentDB 8.0.2). Weitere Informationen zu Planer-Versionen finden Sie unter. Abfrageplaner v3

Amazon DocumentDB kann keinen Teilindex oder einen Multikey-Index (Array) verwenden, um das vorsortierte Präfix für eine inkrementelle Sortierung bereitzustellen. In diesen Fällen greift Amazon DocumentDB auf eine vollständige Sortierung zurück. Dies beeinträchtigt niemals die Richtigkeit der Ergebnisse.

Überprüfung einer inkrementellen Sortierung mit **`explain()`**

Wird verwendet`explain()`, um zu bestätigen, dass eine Abfrage eine inkrementelle Sortierung verwendet. Im erfolgreichen Plan erscheint eine inkrementelle Sortierung als `SORT` Stufe, die ein `presortedKeys` Array enthält, das die führenden Sortierfelder auflistet, die der Index bereits sortiert hat. Eine vollständige Sortierung wird als `SORT` Stufe ohne `presortedKeys` Array angezeigt. Das

`presortedKeys` Feld ist eine DocumentDB-specific Amazon-Ergänzung zur `explain()` Ausgabe und erscheint nicht in den Erläuterungsplänen von MongoDB.

```
db.orders.createIndex({ customerId: 1 })

db.orders.find({}).sort({ customerId: 1, orderDate: -1 }).explain()
```

```
{
    "stage": "SORT",
    "sortPattern": {
        "customerId": 1,
        "orderDate": -1
    },
    "presortedKeys": [
        "customerId"
    ],
    "inputStage": {
        "stage": "IXSCAN",
        "indexName": "customerId_1",
        "direction": "forward"
    }
}
```

$sortArray

Neu ab Version 8.0

Wird von Elastic Cluster nicht unterstützt.

Der `$sortArray` Operator in Amazon DocumentDB sortiert die Elemente eines Arrays. Bei Arrays mit Skalarwerten wird das Array nach Elementwerten sortiert. Bei Dokumentenarrays wird das Array nach dem oder den angegebenen Feldern sortiert. Der Operator gibt das sortierte Array in aufsteigender (1) oder absteigender (-1) Reihenfolge zurück.

> **Note**
>
> `$sortArray`wird in der `$project` Phase der Aggregationspipeline mit Query Planner Version 3 unterstützt.

Syntax

```
{ $sortArray: { input: <array>, sortBy: <sort specification> } }
```

Parameter

- `input`: Das zu sortierende Array.
- `sortBy`: Die Sortierreihenfolge. Geben Sie für eine Reihe von Skalarwerten die Werte 1 für aufsteigend oder `-1` absteigend an. Geben Sie für eine Reihe von Dokumenten ein Dokument des Formulars `{ field: 1 }` oder `{ field: -1 }` für jedes Feld an, nach dem sortiert werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$sortArray` Operator verwendet wird, um die Elemente eines Arrays in aufsteigender Reihenfolge zu sortieren.

Erstellen Sie Beispieldokumente

```
db.miles.insertMany([
  { "_id" : 1, "member_since" : ISODate("1987-01-01T00:00:00Z"), "credit_card" : false,
 "flight_miles" : [ 1205, 2560, 880 ]},
  { "_id" : 2, "member_since" : ISODate("1982-01-01T00:00:00Z"), "credit_card" : true,
 "flight_miles" : [ 1205, 2560, 890, 2780]},
  { "_id" : 3, "member_since" : ISODate("1999-01-01T00:00:00Z"), "credit_card" : true,
 "flight_miles" : [ 1205, 880]}
]);
```

Beispiel für eine Abfrage

```
db.miles.aggregate([
  {
    $project: {
      _id: 1,
      member_since: 1,
      credit_card: 1,
      sorted_flight_miles: { $sortArray: { input: "$flight_miles", sortBy: 1 } }
    }
  }
]);
```

Ausgabe

```
{ "_id" : 1, "member_since" : ISODate("1987-01-01T00:00:00Z"), "credit_card" : false,
 "sorted_flight_miles" : [ 880, 1205, 2560 ] }
{ "_id" : 2, "member_since" : ISODate("1982-01-01T00:00:00Z"), "credit_card" : true,
 "sorted_flight_miles" : [ 890, 1205, 2560, 2780 ] }
{ "_id" : 3, "member_since" : ISODate("1999-01-01T00:00:00Z"), "credit_card" : true,
 "sorted_flight_miles" : [ 880, 1205 ] }
```

In diesem Beispiel sortiert der `$sortArray` Operator das `flight_miles` Array in aufsteigender Reihenfolge. Das resultierende `sorted_flight_miles` Feld in der Ausgabe zeigt die Elemente des Arrays in sortierter Reihenfolge.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$sortArray` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

Hier ist ein Beispiel für die Verwendung des `$sortArray` Operators in einer Node.js Anwendung:

```
const { MongoClient } = require('mongodb');

async function sortArray() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('miles');

  const result = await collection.aggregate([
    {
      $project: {
        _id: 1,
        member_since: 1,
        credit_card: 1,
        sorted_flight_miles: { $sortArray: { input: '$flight_miles', sortBy: 1 } }
      }
    }
  ]).toArray();

  console.log(result);
  client.close();
}
```

```
sortArray();
```

Python

Hier ist ein Beispiel für die Verwendung des `$sortArray` Operators in einer Python-Anwendung:

```
from pymongo import MongoClient

def sort_array():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.miles

    result = list(collection.aggregate([
        {
            '$project': {
                '_id': 1,
                'member_since': 1,
                'credit_card': 1,
                'sorted_flight_miles': { '$sortArray': { 'input': '$flight_miles',
 'sortBy': 1 } }
            }
        }
    ]))

    print(result)
    client.close()

sort_array()
```

$ sortieren ByCount

Neu ab Version 8.0.1.

Die `$sortByCount` Aggregationsphase in Amazon DocumentDB gruppiert eingehende Dokumente anhand des Werts eines angegebenen Ausdrucks, berechnet dann die Anzahl der Dokumente in jeder einzelnen Gruppe und sortiert die Ergebnisse nach Anzahl in absteigender Reihenfolge. Dies entspricht einer `$group` Stufe mit einem `$sum` Akkumulator, auf den eine `$sort` Stufe im Zählfeld folgt.

Parameter

- `expression`: Der Ausdruck, nach dem gruppiert werden soll. Dies kann ein Feldpfad (mit Präfix$) oder ein beliebiger gültiger Aggregationsausdruck sein.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie die `$sortByCount` Bühne verwendet wird, um Dokumente nach einem Kategoriefeld zu zählen und zu sortieren.

Erstellen Sie Beispieldokumente

```
db.products.insertMany([
  { name: "Widget", category: "electronics" },
  { name: "Gadget", category: "electronics" },
  { name: "Doohickey", category: "electronics" },
  { name: "Apple", category: "food" },
  { name: "Banana", category: "food" },
  { name: "Hammer", category: "tools" }
]);
```

Beispiel für eine Abfrage

```
db.products.aggregate([
  { $sortByCount: "$category" }
]);
```

Ausgabe

```
[
  { "_id": "electronics", "count": 3 },
  { "_id": "food", "count": 2 },
  { "_id": "tools", "count": 1 }
]
```

Die Ergebnisse werden automatisch `count` in absteigender Reihenfolge sortiert.

Codebeispiele

Um ein Codebeispiel für die Verwendung der `$sortByCount` Stufe anzuzeigen, wählen Sie den Tab für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  try {
    await client.connect();
    const db = client.db('test');
    const collection = db.collection('products');

    const result = await collection.aggregate([
      { $sortByCount: "$category" }
    ]).toArray();

    console.log(result);
  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    try:
        db = client['test']
        collection = db['products']

        result = list(collection.aggregate([
            { '$sortByCount': '$category' }
        ]))
```

```
        print(result)
    finally:
        client.close()

example()
```

$split

Der `$split` Aggregationsoperator in Amazon DocumentDB wird verwendet, um eine Zeichenfolge auf der Grundlage eines angegebenen Trennzeichens in ein Array von Teilzeichenfolgen aufzuteilen. Dies kann nützlich sein, um komplexe Zeichenkettenfelder zu analysieren und einzelne Komponenten für die weitere Verarbeitung zu extrahieren.

Parameter

- `string`: Die Zeichenfolge, die aufgeteilt werden soll.
- `delimiter`: Das Zeichen oder die Zeichenfolge, das zum Teilen der Eingabezeichenfolge verwendet wurde.

Beispiel (MongoDB Shell)

In diesem Beispiel trennen wir `$split` die Komponenten eines Feldes „Desk“ in ein Array, um die Verarbeitung der Daten zu vereinfachen.

Erstellen Sie Beispieldokumente

```
db.people.insertMany([
  { "_id": 1, "Desk": "Düsseldorf-BVV-021" },
  { "_id": 2, "Desk": "Munich-HGG-32a" },
  { "_id": 3, "Desk": "Cologne-ayu-892.50" },
  { "_id": 4, "Desk": "Dortmund-Hop-78" }
]);
```

Beispiel für eine Abfrage

```
db.people.aggregate([
  { $project: { parts: { $split: ["$Desk", "-"] } } }
]);
```

Ausgabe

```
{ "_id" : 1, "parts" : [ "Düsseldorf", "BVV", "021" ] }
{ "_id" : 2, "parts" : [ "Munich", "HGG", "32a" ] }
{ "_id" : 3, "parts" : [ "Cologne", "ayu", "892.50" ] }
{ "_id" : 4, "parts" : [ "Dortmund", "Hop", "78" ] }
```

Die Ausgabe von `$split` erstellt ein Array, das in der Anwendung verwendet werden kann, um die Informationen für die Mitarbeiter anzuzeigen.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$split` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const result = await db.collection('people').aggregate([
    { $project: { parts: { $split: ['$Desk', '-'] } } }
  ]).toArray();
  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
```

```
    result = list(db.people.aggregate([
        { '$project': { 'parts': { '$split': ['$Desk', '-'] } } }
    ]))
    print(result)
    client.close()

example()
```

$sqrt

Neu ab Version 4.0.

Der `$sqrt` Operator in Amazon DocumentDB wird verwendet, um die Quadratwurzel einer Zahl zu berechnen.

Parameter

- `expression`: Das Argument kann ein beliebiger gültiger Ausdruck sein, solange es zu einer nicht negativen Zahl aufgelöst wird.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung des `$sqrt` Operators zur Berechnung der Quadratwurzel einer Zahl.

Erstellen Sie Beispieldokumente

```
db.numbers.insertMany([
  { "_id": 1, "number": 16 },
  { "_id": 2, "number": 36 },
  { "_id": 3, "number": 64 }
]);
```

Beispiel für eine Abfrage

```
db.numbers.aggregate([
  { $project: {
    "_id": 1,
    "square_root": { $sqrt: "$number" }
  }}
```

```
]);
```

Ausgabe

```
[
  { _id: 1, square_root: 4 },
  { _id: 2, square_root: 6 },
  { _id: 3, square_root: 8 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$sqrt` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  try {
      await client.connect();

      const db = client.db('test');
      const collection = db.collection('numbers');

      const pipeline = [
        {
          $project: {
            _id: 1,
            square_root: { $sqrt: '$number' }
          }
        }
      ];

      const results = await collection.aggregate(pipeline).toArray();

      console.dir(results, { depth: null });
```

```
    } finally {
      await client.close();
    }
}

example().catch(console.error);
```

Python

```
from pymongo import MongoClient

def example():

  client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

  try:
      db = client.test
      collection = db.numbers

      pipeline = [
          {
              "$project": {
                  "_id": 1,
                  "square_root": {
                    "$sqrt": "$number"
                  }
              }
          }
      ]

      results = collection.aggregate(pipeline)

      for doc in results:
          print(doc)

  except Exception as e:
      print(f"An error occurred: {e}")

  finally:
      client.close()
```

```
example()
```

$ std DevPop

Neu ab Version 8.0.1.

Der `$stdDevPop` Operator in Amazon DocumentDB berechnet die Populationsstandardabweichung numerischer Werte. Als Akkumulator berechnet er in der `$group` Phase einer Aggregationspipeline die Standardabweichung der Grundgesamtheiten für Dokumente innerhalb einer Gruppe. Als Ausdruck berechnet er die Standardabweichung der Grundgesamtheit einer Reihe von Zahlen. Bei der Standardabweichung der Grundgesamtheit wird N als Divisor verwendet (nicht N-1). Non-numeric Werte werden ignoriert. Wenn es keine numerischen Werte gibt, wird zurückgegeben`null`. Wenn es nur einen numerischen Wert gibt, wird er zurückgegeben`0`.

Parameter

- `expression`: Ein Ausdruck, der in einen numerischen Wert oder ein Array numerischer Werte aufgelöst wird.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$stdDevPop` Operator verwendet wird, um die Populationsstandardabweichung der Punktzahlen pro Proband zu berechnen.

Erstellen Sie Beispieldokumente

```
db.scores.insertMany([
  { subject: "math", score: 60 },
  { subject: "math", score: 75 },
  { subject: "math", score: 85 },
  { subject: "math", score: 92 },
  { subject: "math", score: 78 },
  { subject: "science", score: 55 },
  { subject: "science", score: 70 },
  { subject: "science", score: 82 },
  { subject: "science", score: 91 },
  { subject: "science", score: 67 }
]);
```

Beispiel für eine Abfrage

```
db.scores.aggregate([
  { $group: {
      _id: "$subject",
      stdDev: { $stdDevPop: "$score" }
    }}
]);
```

Ausgabe

```
[
  { "_id": "math", "stdDev": 10.75174404457249 },
  { "_id": "science", "stdDev": 12.441864811996632 }
]
```

Beispiel für die Verwendung von Ausdrücken (MongoDB Shell)

Der `$stdDevPop` Operator kann auch als Ausdruck innerhalb einer `$project` Phase verwendet werden, um die Populationsstandardabweichung eines Array-Feldes zu berechnen.

Erstellen Sie Beispieldokumente

```
db.experiments.insertMany([
  { _id: 1, measurements: [10, 12, 14, 16, 18] },
  { _id: 2, measurements: [5, 5, 5, 5, 5] },
  { _id: 3, measurements: [2, 4, 6, 8, 10] }
]);
```

Beispiel für eine Abfrage

```
db.experiments.aggregate([
  { $project: {
      stdDev: { $stdDevPop: "$measurements" }
    }}
]);
```

Ausgabe

```
[
  { "_id": 1, "stdDev": 2.8284271247461903 },
  { "_id": 2, "stdDev": 0 },
  { "_id": 3, "stdDev": 2.8284271247461903 }
```

```
]
```

Beispiel für die Verwendung von Fensteroperatoren (MongoDB Shell)

Neu ab Version 8.0.2.

Der `$stdDevPop` Operator kann auch als Fensterbediener auf der `$setWindowFields` Bühne verwendet werden. In diesem Zusammenhang wird die Standardabweichung der Grundgesamtheit des angegebenen Ausdrucks für die Dokumente in jedem Fenster zurückgegeben. Sie geben den Operator unter dem `output` Feld an und definieren optional die Fenstergrenzen mit einem `window` Dokument.

Erstellen Sie Beispieldokumente

```
db.sensorData.insertMany([
  { _id: 1, sensor: "A", time: 1, reading: 4 },
  { _id: 2, sensor: "A", time: 2, reading: 10 },
  { _id: 3, sensor: "A", time: 3, reading: 10 },
  { _id: 4, sensor: "B", time: 1, reading: 10 },
  { _id: 5, sensor: "B", time: 2, reading: 20 }
]);
```

Beispiel für eine Abfrage

Das folgende Beispiel partitioniert die Dokumente nach`sensor`, sortiert jede Partition nach `time` und gibt die aktuelle Standardabweichung der Population `reading` vom Anfang der Partition bis zum aktuellen Dokument zurück.

```
db.sensorData.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$sensor",
      sortBy: { time: 1 },
      output: {
        runningStdDev: {
          $stdDevPop: "$reading",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "sensor": "A", "time": 1, "reading": 4, "runningStdDev": 0 },
  { "_id": 2, "sensor": "A", "time": 2, "reading": 10, "runningStdDev": 3 },
  { "_id": 3, "sensor": "A", "time": 3, "reading": 10, "runningStdDev":
 2.8284271247461903 },
  { "_id": 4, "sensor": "B", "time": 1, "reading": 10, "runningStdDev": 0 },
  { "_id": 5, "sensor": "B", "time": 2, "reading": 20, "runningStdDev": 5 }
]
```

Jedes Dokument wird um die Standardabweichung der Grundgesamtheit `reading` innerhalb seiner Partition bis einschließlich des aktuellen Dokuments erweitert. `runningStdDev` Wenn das Fenster einen einzelnen Wert enthält, wird `$stdDevPop` zurückgegeben`0`.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$stdDevPop` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten. Die folgenden Beispiele zeigen die Verwendung von Akkumulatoren (in`$group`), Ausdrücken (in`$project`) und Fensteroperatoren (in`$setWindowFields`):

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const uri = 'mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false';
  const client = new MongoClient(uri);

  try {
    await client.connect();
    const db = client.db('test');

    // Accumulator usage: stdDevPop across grouped documents
    const scores = db.collection('scores');
    await scores.insertMany([
      { subject: "math", score: 60 },
      { subject: "math", score: 75 },
      { subject: "math", score: 85 },
```

```
    { subject: "math", score: 92 },
    { subject: "math", score: 78 },
    { subject: "science", score: 55 },
    { subject: "science", score: 70 },
    { subject: "science", score: 82 },
    { subject: "science", score: 91 },
    { subject: "science", score: 67 }
]);
const accumulatorResult = await scores.aggregate([
    { $group: {
        _id: "$subject",
        stdDev: { $stdDevPop: "$score" }
      }}
]).toArray();
console.log('Accumulator result:', accumulatorResult);

// Expression usage: stdDevPop of an array field
const experiments = db.collection('experiments');
await experiments.insertMany([
    { _id: 1, measurements: [10, 12, 14, 16, 18] },
    { _id: 2, measurements: [5, 5, 5, 5, 5] },
    { _id: 3, measurements: [2, 4, 6, 8, 10] }
]);
const expressionResult = await experiments.aggregate([
    { $project: {
        stdDev: { $stdDevPop: "$measurements" }
      }}
]).toArray();
console.log('Expression result:', expressionResult);

// Window operator usage: running stdDevPop within each partition
const sensorData = db.collection('sensorData');
await sensorData.insertMany([
    { _id: 1, sensor: "A", time: 1, reading: 4 },
    { _id: 2, sensor: "A", time: 2, reading: 10 },
    { _id: 3, sensor: "A", time: 3, reading: 10 },
    { _id: 4, sensor: "B", time: 1, reading: 10 },
    { _id: 5, sensor: "B", time: 2, reading: 20 }
]);
const windowResult = await sensorData.aggregate([
    {
      $setWindowFields: {
        partitionBy: "$sensor",
        sortBy: { time: 1 },
```

```
          output: {
            runningStdDev: {
              $stdDevPop: "$reading",
              window: { documents: ["unbounded", "current"] }
            }
          }
        }
      }
    ]).toArray();
    console.log('Window result:', windowResult);

  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    try:
        db = client['test']

        # Accumulator usage: stdDevPop across grouped documents
        scores = db['scores']
        scores.insert_many([
            { 'subject': 'math', 'score': 60 },
            { 'subject': 'math', 'score': 75 },
            { 'subject': 'math', 'score': 85 },
            { 'subject': 'math', 'score': 92 },
            { 'subject': 'math', 'score': 78 },
            { 'subject': 'science', 'score': 55 },
            { 'subject': 'science', 'score': 70 },
            { 'subject': 'science', 'score': 82 },
            { 'subject': 'science', 'score': 91 },
            { 'subject': 'science', 'score': 67 }
```

```
        ])
        accumulator_result = list(scores.aggregate([
            { '$group': {
                '_id': '$subject',
                'stdDev': { '$stdDevPop': '$score' }
            }}
        ]))
        print('Accumulator result:', accumulator_result)

        # Expression usage: stdDevPop of an array field
        experiments = db['experiments']
        experiments.insert_many([
            { '_id': 1, 'measurements': [10, 12, 14, 16, 18] },
            { '_id': 2, 'measurements': [5, 5, 5, 5, 5] },
            { '_id': 3, 'measurements': [2, 4, 6, 8, 10] }
        ])
        expression_result = list(experiments.aggregate([
            { '$project': {
                'stdDev': { '$stdDevPop': '$measurements' }
            }}
        ]))
        print('Expression result:', expression_result)

        # Window operator usage: running stdDevPop within each partition
        sensor_data = db['sensorData']
        sensor_data.insert_many([
            { '_id': 1, 'sensor': 'A', 'time': 1, 'reading': 4 },
            { '_id': 2, 'sensor': 'A', 'time': 2, 'reading': 10 },
            { '_id': 3, 'sensor': 'A', 'time': 3, 'reading': 10 },
            { '_id': 4, 'sensor': 'B', 'time': 1, 'reading': 10 },
            { '_id': 5, 'sensor': 'B', 'time': 2, 'reading': 20 }
        ])
        window_result = list(sensor_data.aggregate([
            {
                '$setWindowFields': {
                    'partitionBy': '$sensor',
                    'sortBy': { 'time': 1 },
                    'output': {
                        'runningStdDev': {
                            '$stdDevPop': '$reading',
                            'window': { 'documents': ['unbounded', 'current'] }
                        }
                    }
                }
```

```
            }
        ]))
        print('Window result:', window_result)

    finally:
        client.close()

example()
```

$ std DevSamp

Neu ab Version 8.0.1.

Der `$stdDevSamp` Operator in Amazon DocumentDB berechnet die Standardabweichung der Stichprobe numerischer Werte. Als Akkumulator berechnet er in der `$group` Phase einer Aggregationspipeline die Standardabweichung der Stichprobe für Dokumente innerhalb einer Gruppe. Als Ausdruck berechnet er die Standardabweichung der Stichprobe einer Reihe von Zahlen. N-1 Als Divisor wird die Standardabweichung der Stichprobe verwendet (Bessel-Korrektur). Non-numeric Werte werden ignoriert. Wenn es weniger als zwei numerische Werte gibt, wird zurückgegeben`null`.

Parameter

- `expression`: Ein Ausdruck, der in einen numerischen Wert oder ein Array numerischer Werte aufgelöst wird.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$stdDevSamp` Operator verwendet wird, um die Stichprobenstandardabweichung der Ergebnisse pro Proband zu berechnen.

Erstellen Sie Beispieldokumente

```
db.scores.insertMany([
  { subject: "math", score: 80 },
  { subject: "math", score: 90 },
  { subject: "math", score: 85 },
  { subject: "math", score: 95 },
  { subject: "science", score: 70 },
  { subject: "science", score: 75 },
```

```
  { subject: "science", score: 80 },
  { subject: "science", score: 85 }
]);
```

Beispiel für eine Abfrage

```
db.scores.aggregate([
  { $group: {
      _id: "$subject",
      stdDev: { $stdDevSamp: "$score" }
    }}
]);
```

Ausgabe

```
[
  { "_id": "math", "stdDev": 6.454972243679028 },
  { "_id": "science", "stdDev": 6.454972243679028 }
]
```

Beispiel für die Verwendung von Ausdrücken (MongoDB Shell)

Der `$stdDevSamp` Operator kann auch als Ausdruck innerhalb einer `$project` Stufe verwendet werden, um die Standardabweichung der Stichprobe eines Array-Feldes zu berechnen.

Erstellen Sie Beispieldokumente

```
db.experiments.insertMany([
  { _id: 1, measurements: [10, 12, 14, 16, 18] },
  { _id: 2, measurements: [5, 5, 5, 5, 5] },
  { _id: 3, measurements: [2, 4, 6, 8, 10] }
]);
```

Beispiel für eine Abfrage

```
db.experiments.aggregate([
  { $project: {
      stdDev: { $stdDevSamp: "$measurements" }
    }}
]);
```

Ausgabe

```
[
  { "_id": 1, "stdDev": 3.1622776601683795 },
  { "_id": 2, "stdDev": 0 },
  { "_id": 3, "stdDev": 3.1622776601683795 }
]
```

Beispiel für die Verwendung von Fensteroperatoren (MongoDB Shell)

Neu ab Version 8.0.2.

Der `$stdDevSamp` Operator kann auch als Fensterbediener auf der `$setWindowFields` Bühne verwendet werden. In diesem Zusammenhang gibt es die Standardabweichung der Stichprobe des angegebenen Ausdrucks für die Dokumente in jedem Fenster zurück. Sie geben den Operator unter dem `output` Feld an und definieren optional die Fenstergrenzen mit einem `window` Dokument.

Erstellen Sie Beispieldokumente

```
db.sensorReadings.insertMany([
  { _id: 1, sensor: "A", time: 1, value: 2 },
  { _id: 2, sensor: "A", time: 2, value: 4 },
  { _id: 3, sensor: "A", time: 3, value: 6 },
  { _id: 4, sensor: "B", time: 1, value: 10 },
  { _id: 5, sensor: "B", time: 2, value: 12 }
]);
```

Beispiel für eine Abfrage

Das folgende Beispiel partitioniert die Dokumente nach`sensor`, sortiert jede Partition nach `time` und gibt die Standardabweichung der laufenden Stichprobe `value` vom Anfang der Partition bis zum aktuellen Dokument zurück.

```
db.sensorReadings.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$sensor",
      sortBy: { time: 1 },
      output: {
        runningStdDev: {
          $stdDevSamp: "$value",
          window: { documents: ["unbounded", "current"] }
        }
      }
```

```
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "sensor": "A", "time": 1, "value": 2, "runningStdDev": null },
  { "_id": 2, "sensor": "A", "time": 2, "value": 4, "runningStdDev":
 1.4142135623730951 },
  { "_id": 3, "sensor": "A", "time": 3, "value": 6, "runningStdDev": 2 },
  { "_id": 4, "sensor": "B", "time": 1, "value": 10, "runningStdDev": null },
  { "_id": 5, "sensor": "B", "time": 2, "value": 12, "runningStdDev":
 1.4142135623730951 }
]
```

Jedes Dokument wird um die Standardabweichung der Stichprobe `value` innerhalb seiner Partition bis einschließlich des aktuellen Dokuments erweitert. `runningStdDev` Wenn das Fenster weniger als zwei Werte enthält (das erste Dokument in jeder Partition), wird `$stdDevSamp` zurückgegeben`null`.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$stdDevSamp` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten. Die folgenden Beispiele zeigen die Verwendung von Akkumulatoren (in`$group`), Ausdrücken (in`$project`) und Fensteroperatoren (in`$setWindowFields`):

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const uri = 'mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false';
  const client = new MongoClient(uri);

  try {
    await client.connect();
    const db = client.db('test');
```

```
// Accumulator usage: stdDevSamp across grouped documents
const scores = db.collection('scores');
await scores.insertMany([
  { subject: "math", score: 80 },
  { subject: "math", score: 90 },
  { subject: "math", score: 85 },
  { subject: "math", score: 95 },
  { subject: "science", score: 70 },
  { subject: "science", score: 75 },
  { subject: "science", score: 80 },
  { subject: "science", score: 85 }
]);
const accumulatorResult = await scores.aggregate([
  { $group: {
      _id: "$subject",
      stdDev: { $stdDevSamp: "$score" }
    }}
]).toArray();
console.log('Accumulator result:', accumulatorResult);

// Expression usage: stdDevSamp of an array field
const experiments = db.collection('experiments');
await experiments.insertMany([
  { _id: 1, measurements: [10, 12, 14, 16, 18] },
  { _id: 2, measurements: [5, 5, 5, 5, 5] },
  { _id: 3, measurements: [2, 4, 6, 8, 10] }
]);
const expressionResult = await experiments.aggregate([
  { $project: {
      stdDev: { $stdDevSamp: "$measurements" }
    }}
]).toArray();
console.log('Expression result:', expressionResult);

// Window operator usage: running stdDevSamp within each partition
const sensorReadings = db.collection('sensorReadings');
await sensorReadings.insertMany([
  { _id: 1, sensor: "A", time: 1, value: 2 },
  { _id: 2, sensor: "A", time: 2, value: 4 },
  { _id: 3, sensor: "A", time: 3, value: 6 },
  { _id: 4, sensor: "B", time: 1, value: 10 },
  { _id: 5, sensor: "B", time: 2, value: 12 }
]);
const windowResult = await sensorReadings.aggregate([
```

```
      {
        $setWindowFields: {
          partitionBy: "$sensor",
          sortBy: { time: 1 },
          output: {
            runningStdDev: {
              $stdDevSamp: "$value",
              window: { documents: ["unbounded", "current"] }
            }
          }
        }
      }
    ]).toArray();
    console.log('Window result:', windowResult);

  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    try:
        db = client['test']

        # Accumulator usage: stdDevSamp across grouped documents
        scores = db['scores']
        scores.insert_many([
            { 'subject': 'math', 'score': 80 },
            { 'subject': 'math', 'score': 90 },
            { 'subject': 'math', 'score': 85 },
            { 'subject': 'math', 'score': 95 },
            { 'subject': 'science', 'score': 70 },
            { 'subject': 'science', 'score': 75 },
```

```
        { 'subject': 'science', 'score': 80 },
        { 'subject': 'science', 'score': 85 }
    ])
    accumulator_result = list(scores.aggregate([
        { '$group': {
            '_id': '$subject',
            'stdDev': { '$stdDevSamp': '$score' }
        }}
    ]))
    print('Accumulator result:', accumulator_result)

    # Expression usage: stdDevSamp of an array field
    experiments = db['experiments']
    experiments.insert_many([
        { '_id': 1, 'measurements': [10, 12, 14, 16, 18] },
        { '_id': 2, 'measurements': [5, 5, 5, 5, 5] },
        { '_id': 3, 'measurements': [2, 4, 6, 8, 10] }
    ])
    expression_result = list(experiments.aggregate([
        { '$project': {
            'stdDev': { '$stdDevSamp': '$measurements' }
        }}
    ]))
    print('Expression result:', expression_result)

    # Window operator usage: running stdDevSamp within each partition
    sensor_readings = db['sensorReadings']
    sensor_readings.insert_many([
        { '_id': 1, 'sensor': 'A', 'time': 1, 'value': 2 },
        { '_id': 2, 'sensor': 'A', 'time': 2, 'value': 4 },
        { '_id': 3, 'sensor': 'A', 'time': 3, 'value': 6 },
        { '_id': 4, 'sensor': 'B', 'time': 1, 'value': 10 },
        { '_id': 5, 'sensor': 'B', 'time': 2, 'value': 12 }
    ])
    window_result = list(sensor_readings.aggregate([
        {
            '$setWindowFields': {
                'partitionBy': '$sensor',
                'sortBy': { 'time': 1 },
                'output': {
                    'runningStdDev': {
                        '$stdDevSamp': '$value',
                        'window': { 'documents': ['unbounded', 'current'] }
                    }
```

```
                    }
                }
            }
        ]))
        print('Window result:', window_result)

    finally:
        client.close()

example()
```

$str LenBytes

Der `$strLenBytes` Operator in Amazon DocumentDB wird verwendet, um die Länge einer Zeichenfolge in Byte zu bestimmen. Dies ist nützlich, wenn Sie die Speichergröße eines Zeichenkettenfeldes verstehen müssen, insbesondere wenn Sie es mit Unicode-Zeichen zu tun haben, die mehr als ein Byte pro Zeichen verwenden können.

Parameter

- `expression`: Der Zeichenkettenausdruck, dessen Länge berechnet werden soll.

Beispiel (MongoDB Shell)

Dieses Beispiel zeigt, wie der `$strLenBytes` Operator verwendet wird, um die Länge von Zeichenkettenfeldern in Byte zu berechnen.

Erstellen Sie Beispieldokumente

```
db.people.insertMany([
  { "_id": 1, "Desk": "Düsseldorf-BVV-021" },
  { "_id": 2, "Desk": "Munich-HGG-32a" },
  { "_id": 3, "Desk": "Cologne-ayu-892.50" },
  { "_id": 4, "Desk": "Dortmund-Hop-78" }
]);
```

Beispiel für eine Abfrage

```
db.people.aggregate([
  {
```

```
    $project: {
      "Desk": 1,
      "length": { $strLenBytes: "$Desk" }
    }
  }
])
```

Ausgabe

```
{ "_id" : 1, "Desk" : "Düsseldorf-BVV-021", "length" : 19 }
{ "_id" : 2, "Desk" : "Munich-HGG-32a", "length" : 14 }
{ "_id" : 3, "Desk" : "Cologne-ayu-892.50", "length" : 18 }
{ "_id" : 4, "Desk" : "Dortmund-Hop-78", "length" : 15 }
```

Beachten Sie, dass die Länge der Zeichenfolge „Düsseldorf-BVV-021“ 19 Byte beträgt, was sich von der Anzahl der Codepunkte (18) unterscheidet, da das Unicode-Zeichen „Ü“ 2 Byte belegt.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$strLenBytes` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('people');

  const result = await collection.aggregate([
    {
      $project: {
        "Desk": 1,
        "length": { $strLenBytes: "$Desk" }
      }
    }
  ]).toArray();
```

```
  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.people

    result = list(collection.aggregate([
        {
            '$project': {
                "Desk": 1,
                "length": { "$strLenBytes": "$Desk" }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$strLenCP

Der `$strLenCP` Operator in Amazon DocumentDB wird verwendet, um die Länge eines Zeichenkettenausdrucks in Codepunkten (Unicode-Zeichen) zu bestimmen. Dies ist nützlich, wenn Sie die Anzahl der Zeichen in einer Zeichenfolge und nicht die Anzahl der Byte kennen müssen.

Parameter

- `expression`: Der Zeichenkettenausdruck, für den die Länge in Codepunkten zurückgegeben werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung des `$strLenCP` Operators zur Bestimmung der Länge von Zeichenketten mit Unicode-Zeichen.

Erstellen Sie Beispieldokumente

```
db.people.insertMany([
  { "_id": 1, "Desk": "Düsseldorf-BVV-021" },
  { "_id": 2, "Desk": "Munich-HGG-32a" },
  { "_id": 3, "Desk": "Cologne-ayu-892.50" },
  { "_id": 4, "Desk": "Dortmund-Hop-78" }
])
```

Beispiel für eine Abfrage

```
db.people.aggregate([
  {
    $project: {
      "Desk": 1,
      "length": { $strLenCP: "$Desk" }
    }
  }
])
```

Ausgabe

```
{ "_id" : 1, "Desk" : "Düsseldorf-BVV-021", "length" : 18 }
{ "_id" : 2, "Desk" : "Munich-HGG-32a", "length" : 14 }
{ "_id" : 3, "Desk" : "Cologne-ayu-892.50", "length" : 18 }
{ "_id" : 4, "Desk" : "Dortmund-Hop-78", "length" : 15 }
```

Beachten Sie den Unterschied in der Längenmessung für die Zeichenfolge „Düsseldorf-BVV-021“, die ein Unicode-Zeichen (Ü) enthält. Der `$strLenCP` Operator zählt die Anzahl der Unicode-Zeichen korrekt, während der `$strLenBytes` Operator die Anzahl der Byte zählt.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$strLenCP` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

Hier ist ein Beispiel für die Verwendung des `$strLenCP` Operators in einer Node.js Anwendung mit dem MongoDB-Treiber:

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('people');

  const result = await collection.aggregate([
    {
      $project: {
        "Desk": 1,
        "length": { $strLenCP: "$Desk" }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

Hier ist ein Beispiel für die Verwendung des `$strLenCP` Operators in einer Python-Anwendung mit dem PyMongo Treiber:

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.people
```

```
    result = list(collection.aggregate([
        {
            '$project': {
                "Desk": 1,
                "length": { "$strLenCP": "$Desk" }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$strcasecmp

Der `$strcasecmp` Operator in Amazon DocumentDB führt einen Vergleich zwischen zwei Zeichenketten durch, bei dem die Groß- und Kleinschreibung nicht beachtet wird. Er gibt einen Integer-Wert zurück, der den lexikografischen Vergleich der beiden Eingabezeichenfolgen angibt, wobei Unterschiede zwischen Groß- und Kleinschreibung ignoriert werden.

Parameter

- `string1`: Die erste Zeichenfolge, die verglichen werden soll.
- `string2`: Die zweite zu vergleichende Zeichenfolge.

Beispiel (MongoDB Shell)

Dieses Beispiel zeigt, wie der `$strcasecmp` Operator verwendet wird, um Zeichenketten mit Schreibtischstandorten in einer `people` Sammlung zu vergleichen, wobei Unterschiede zwischen Groß- und Kleinschreibung ignoriert werden.

Erstellen Sie Beispieldokumente

```
db.people.insertMany([
  { "_id": 1, "Desk": "mke233-wi" },
  { "_id": 2, "Desk": "MKE233-WI" },
  { "_id": 3, "Desk": "mke233-wi" }
]);
```

Beispiel für eine Abfrage

```
db.people.aggregate([
  {
    $project: {
      item: 1,
      compare: { $strcasecmp: ["$Desk", "mke233-wi"] }
    }
  }
]);
```

Ausgabe

```
{ "_id" : 1, "compare" : 0 }
{ "_id" : 2, "compare" : 0 }
{ "_id" : 3, "compare" : 0 }
```

Die Ausgabe zeigt, dass der Vergleich zwischen dem "Desk" Feld und der Zeichenfolge 0 für alle drei Dokumente "mke233-wi" zurückgegeben wird, was bedeutet, dass die Zeichenketten identisch sind, wenn die Groß- und Kleinschreibung ignoriert wird.

Codebeispiele

Um ein Codebeispiel für die Verwendung des $strcasecmp Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('people');

  const result = await collection.aggregate([
    {
      $project: {
        item: 1,
```

```
        compare: { $strcasecmp: ["$Desk", "mke233-wi"] }
      }
    }
  ]).toArray();

  console.log(result);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.people

    result = list(collection.aggregate([
        {
            '$project': {
                'item': 1,
                'compare': { '$strcasecmp': ['$Desk', 'mke233-wi'] }
            }
        }
    ]))

    print(result)

    client.close()

example()
```

$substr

Der `$substr` Operator in Amazon DocumentDB wird verwendet, um eine Teilzeichenfolge aus einer bestimmten Zeichenfolge zu extrahieren. Dies ist besonders nützlich, wenn Sie Teilzeichenfolgen

definieren müssen, die auf einem Zeichenbereich und nicht auf einem Bytebereich basieren. Dies ist besonders wichtig, wenn es sich um Unicode-Zeichenketten handelt, bei denen die Anzahl der zur Darstellung eines Zeichens verwendeten Byte variieren kann.

Parameter

- `string`: Die Eingabezeichenfolge, aus der die Teilzeichenfolge extrahiert werden soll.
- `start`: Die Startposition (auf Null basierend) der Teilzeichenfolge, die extrahiert werden soll. Kann ein nicht negativer Integer-Ausdruck sein.
- `length`: Die Anzahl der Zeichen in der extrahierten Teilzeichenfolge. Kann ein nicht negativer Integer-Ausdruck sein.

Beispiel (MongoDB Shell)

In diesem Beispiel demonstrieren wir die Verwendung von, `$substr` um die Abkürzung des Bundesstaates aus dem Schreibtischstandort eines Mitarbeiters zu extrahieren.

Erstellen Sie Beispieldokumente

```
db.people.insertMany([
  { "_id": 1, "Desk": "Düsseldorf-NRW-021" },
  { "_id": 2, "Desk": "Bremerhaven-HBB-32a" },
  { "_id": 3, "Desk": "Norderstedt-SHH-892.50" },
  { "_id": 4, "Desk": "Brandenburg-BBB-78" }
]);
```

Beispiel für eine Abfrage

```
db.people.aggregate([
  {
    $project: {
      "state": { $substr: ["$Desk", 12, 3] }
    }
  }
])
```

Ausgabe

```
{ "_id": 1, "state": "NRW" },
```

```
{ "_id": 2, "state": "HBB" },
{ "_id": 3, "state": "SHH" },
{ "_id": 4, "state": "BBB" }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$substr` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require("mongodb");

async function example() {
  const client = await MongoClient.connect("mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false");
  const db = client.db("test");
  const collection = db.collection("people");

  const result = await collection.aggregate([
    {
      $project: {
        "state": { $substrCP: ["$Desk", 12, 3] }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient("mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false")
    db = client["test"]
```

```
    collection = db["people"]

    result = list(collection.aggregate([
        {
            "$project": {
                "state": { "$substrCP": ["$Desk", 12, 3] }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$substrBytes

Der `$substrBytes` Operator in Amazon DocumentDB wird verwendet, um eine Teilzeichenfolge aus einer Zeichenfolge zu extrahieren, die auf einem bestimmten Bytebereich basiert. Dieser Operator ist nützlich, wenn Sie eine Teilzeichenfolge aus einer Zeichenfolge extrahieren müssen und die Anzahl der Byte, die für die Darstellung der einzelnen Zeichen in der Zeichenfolge erforderlich sind, wichtig ist.

Unlike`$substrCP`, der auf der Anzahl der Unicode-Codepunkte basiert, `$substrBytes` arbeitet mit der Anzahl der Byte, die zur Darstellung der Zeichen in der Zeichenfolge erforderlich sind. Dies kann besonders nützlich sein, wenn Sie mit Zeichenketten arbeiten, die Nicht-ASCII-Zeichen enthalten, da für die Darstellung dieser Zeichen möglicherweise mehr als ein Byte erforderlich ist.

*Hinweis: * `$substr` ist seit Version 3.4 veraltet. `$substr`ist jetzt ein Alias für. `$substrBytes`

Parameter

- `string`: Die Eingabezeichenfolge, aus der die Teilzeichenfolge extrahiert wird.
- `startByte`: Die auf Null basierende Start-Byte-Position der zu extrahierenden Teilzeichenfolge. Ein negativer Wert kann verwendet werden, um eine Position am Ende der Zeichenfolge anzugeben.
- `length`: Die Anzahl der Bytes in der Teilzeichenfolge, die extrahiert werden sollen.

Beispiel (MongoDB Shell)

In diesem Beispiel verwenden wir, `$substrBytes` um eine Teilzeichenfolge aus einer Zeichenfolge zu extrahieren, die Nicht-ASCII-Zeichen enthält.

Erstellen Sie Beispieldokumente

```
db.people.insertMany([
  { "_id": 1, "Desk": "Düsseldorf-NRW-021" },
  { "_id": 2, "Desk": "Bremerhaven-HBB-32a" },
  { "_id": 3, "Desk": "Norderstedt-SHH-892.50" },
  { "_id": 4, "Desk": "Brandenburg-BBB-78" }
]);
```

Beispiel für eine Abfrage

```
db.people.aggregate([
  {
    $project: {
      "state": { $substrBytes: [ "$Desk", 12, 3] }
    }
  }
])
```

Ausgabe

```
{ "_id": 1, "state": "NRW" },
{ "_id": 2, "state": "HBB" },
{ "_id": 3, "state": "SHH" },
{ "_id": 4, "state": "BBB" }
```

In diesem Beispiel verwenden wir, `$substrBytes` um eine 3-Byte-Teilzeichenfolge zu extrahieren, beginnend mit dem 12. Byte des Feldes`Desk`. Auf diese Weise können wir die zweistellige Abkürzung für den Bundesstaat extrahieren, auch wenn die Zeichenfolge Nicht-ASCII-Zeichen enthalten kann.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$substrBytes` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const people = db.collection('people');

  const result = await people.aggregate([
    {
      $project: {
        "state": { $substrBytes: ["$Desk", 12, 3] }
      }
    }
  ]).toArray();

  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    people = db.people

    result = list(people.aggregate([
        {
            '$project': {
                "state": { '$substrBytes': ["$Desk", 12, 3] }
            }
        }
    ]))
```

```
    print(result)
    client.close()

example()
```

$substrCP

Der `$substrCP` Operator in Amazon DocumentDB wird verwendet, um eine Teilzeichenfolge aus einer Zeichenfolge zu extrahieren, wobei die Teilzeichenfolge als Bereich von UTF-8 Codepunkten (CP) angegeben ist. Dieser Operator ist besonders nützlich, wenn Sie mit Unicode-Zeichenketten arbeiten, da Sie damit Teilzeichenfolgen extrahieren können, ohne sich Gedanken über die zugrunde liegende Byte-Darstellung der Zeichen machen zu müssen.

Im Gegensatz zum `$substrBytes` Operator, der mit Bytepositionen arbeitet, arbeitet der `$substrCP` Operator mit Codepunktpositionen. Dies erleichtert die Arbeit mit Zeichenfolgen, die Nicht-ASCII-Zeichen enthalten, da die Anzahl der Codepunkte möglicherweise nicht der Anzahl der Byte oder Zeichen entspricht.

Parameter

- `string`: Die Eingabezeichenfolge, aus der die Teilzeichenfolge extrahiert werden soll.
- `start`: Die Startposition des Codepunkts (auf Null basierend), aus der die Teilzeichenfolge extrahiert werden soll.
- `length`: Die Anzahl der zu extrahierenden Codepunkte.

Beispiel (MongoDB Shell)

In diesem Beispiel verwenden wir den `$substrCP` Operator, um die Abkürzung für das Bundesland aus einer Zeichenfolge zu extrahieren, die den Schreibtischstandort des Mitarbeiters enthält.

Erstellen Sie Beispieldokumente

```
db.people.insert([
  { "_id": 1, "first_name": "Jane", "last_name": "Doe", "Desk": "12 Main St,
 Minneapolis, MN 55401" },
  { "_id": 2, "first_name": "John", "last_name": "Doe", "Desk": "456 Oak Rd, New
 Orleans, LA 70032" },
```

```
  { "_id": 3, "first_name": "Steve", "last_name": "Smith", "Desk": "789 Elm Ln,
 Bakersfield, CA 93263" }
]);
```

Beispiel für eine Abfrage

```
db.people.aggregate([
  {
    $project: {
      "state": { $substrCP: ["$Desk", 25, 2] }
    }
  }
]);
```

Ausgabe

```
{ "_id" : 1, "state" : "MN" }
{ "_id" : 2, "state" : "LA" }
{ "_id" : 3, "state" : "CA" }
```

In diesem Beispiel wissen wir, dass die Abkürzung für das Bundesland am 25. Codepunkt im `Desk` Feld beginnt und 2 Codepunkte lang ist. Mithilfe des `$substrCP` Operators können wir die Abkürzung des Bundesstaates extrahieren, ohne uns um die zugrunde liegende Byte-Darstellung der Zeichenfolge kümmern zu müssen.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$substrCP` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function findStates() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const result = await db.collection('people').aggregate([
    {
      $project: {
```

```
        "state": { $substrCP: ["$Desk", 25, 2] }
      }
    }
  ]).toArray();
  console.log(result);
  client.close();
}

findStates();
```

Python

```
from pymongo import MongoClient

def find_states():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    result = list(db.people.aggregate([
        {
            '$project': {
                'state': { '$substrCP': ['$Desk', 25, 2] }
            }
        }
    ]))
    print(result)
    client.close()

find_states()
```

Sowohl im Beispiel als auch im Node.js Python-Beispiel verwenden wir den `$substrCP` Operator, um die Abkürzung des Bundesstaates aus dem `Desk` Feld zu extrahieren, ähnlich dem MongoDB-Shell-Beispiel.

$subtract

Der `$subtract` Operator in Amazon DocumentDB wird verwendet, um Werte zu subtrahieren. Er kann verwendet werden, um Daten, Zahlen oder eine Kombination aus beidem zu subtrahieren. Dieser Operator ist nützlich, um die Differenz zwischen zwei Daten zu berechnen oder einen Wert von einer Zahl zu subtrahieren.

Parameter

- `expression1`: Der erste Wert, der subtrahiert werden soll.
- `expression2`: Der zweite Wert, von dem subtrahiert werden soll. `<expression1>`

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$subtract` Operator verwendet wird, um die Differenz zwischen zwei Daten zu berechnen.

Erstellen Sie ein Beispieldokument

```
db.dates.insert([
  {
  "_id": 1,
  "startDate": ISODate("2023-01-01T00:00:00Z"),
  "endDate": ISODate("2023-01-05T12:00:00Z")
  }
]);
```

Beispiel für eine Abfrage

```
db.dates.aggregate([
  {
    $project: {
      _id: 1,
      durationDays: {
        $divide: [
          { $subtract: ["$endDate", "$startDate"] },
          1000 * 60 * 60 * 24  // milliseconds in a day
        ]
      }
    }
  }
]);
```

Ausgabe

```
[ { _id: 1, durationDays: 4.5 } ]
```

In diesem Beispiel wird der `$subtract` Operator verwendet, um die Differenz zwischen `$endDate` und `$startDate` in Tagen zu berechnen.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$subtract` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

  try {
    await client.connect();
    const db = client.db('test');
    const collection = db.collection('dates');

    const pipeline = [
      {
        $project: {
          _id: 1,
          durationDays: {
            $divide: [
              { $subtract: ["$endDate", "$startDate"] },
              1000 * 60 * 60 * 24  // Convert milliseconds to days
            ]
          }
        }
      }
    ];

    const results = await collection.aggregate(pipeline).toArray();

    console.dir(results, { depth: null });

  } finally {
    await client.close();
  }
```

```
}

example().catch(console.error);
```

Python

```
from datetime import datetime, timedelta
from pymongo import MongoClient

def example():

    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    try:
        db = client.test
        collection = db.dates

        pipeline = [
            {
                "$project": {
                    "_id": 1,
                    "durationDays": {
                        "$divide": [
                            { "$subtract": ["$endDate", "$startDate"] },
                            1000 * 60 * 60 * 24  # Convert milliseconds to days
                        ]
                    }
                }
            }
        ]

        results = collection.aggregate(pipeline)

        for doc in results:
            print(doc)

    except Exception as e:
        print(f"An error occurred: {e}")

    finally:
        client.close()
```

```
example()
```

$sum

Der `$sum` Operator in Amazon DocumentDB gibt die Summe des angegebenen Ausdrucks für jedes Dokument in einer Gruppe zurück. Es handelt sich um einen Operator für Gruppenakkumulatoren, der typischerweise in der $group-Phase einer Aggregationspipeline verwendet wird, um Summenberechnungen durchzuführen.

Parameter

- `expression`: Der zu summierende numerische Ausdruck. Dies kann ein Feldpfad, ein Ausdruck oder eine Konstante sein.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung des `$sum` Operators zur Berechnung des Gesamtumsatzes für jedes Produkt.

Erstellen Sie Beispieldokumente

```
db.sales.insertMany([
  { product: "abc", price: 10, quantity: 2 },
  { product: "abc", price: 10, quantity: 3 },
  { product: "xyz", price: 20, quantity: 1 },
  { product: "xyz", price: 20, quantity: 5 }
]);
```

Beispiel für eine Abfrage

```
db.sales.aggregate([
  { $group: {
      _id: "$product",
      totalSales: { $sum: { $multiply: [ "$price", "$quantity" ] } }
    }}
]);
```

Ausgabe

```
[
  { "_id": "abc", "totalSales": 50 },
  { "_id": "xyz", "totalSales": 120 }
]
```

Beispiel für die Verwendung von Fensteroperatoren (MongoDB Shell)

Neu ab Version 8.0.2.

Der `$sum` Operator kann auch als Fensterbediener auf der `$setWindowFields` Bühne verwendet werden. In diesem Zusammenhang gibt es die Summe des angegebenen Ausdrucks für die Dokumente in jedem Fenster zurück. Sie geben den Operator unter dem `output` Feld an und definieren optional die Fenstergrenzen mit einem `window` Dokument.

Erstellen Sie Beispieldokumente

```
db.dailySales.insertMany([
  { _id: 1, product: "abc", day: 1, amount: 10 },
  { _id: 2, product: "abc", day: 2, amount: 20 },
  { _id: 3, product: "abc", day: 3, amount: 15 },
  { _id: 4, product: "xyz", day: 1, amount: 30 },
  { _id: 5, product: "xyz", day: 2, amount: 25 }
]);
```

Beispiel für eine Abfrage

Das folgende Beispiel partitioniert die Dokumente nach`product`, sortiert jede Partition nach `day` und gibt eine laufende Summe `amount` vom Anfang der Partition bis zum aktuellen Dokument zurück.

```
db.dailySales.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$product",
      sortBy: { day: 1 },
      output: {
        runningTotal: {
          $sum: "$amount",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
```

```
]);
```

Ausgabe

```
[
  { "_id": 1, "product": "abc", "day": 1, "amount": 10, "runningTotal": 10 },
  { "_id": 2, "product": "abc", "day": 2, "amount": 20, "runningTotal": 30 },
  { "_id": 3, "product": "abc", "day": 3, "amount": 15, "runningTotal": 45 },
  { "_id": 4, "product": "xyz", "day": 1, "amount": 30, "runningTotal": 30 },
  { "_id": 5, "product": "xyz", "day": 2, "amount": 25, "runningTotal": 55 }
]
```

Jedes Dokument wird um die kumulative Summe `amount` innerhalb der Partition bis einschließlich des aktuellen Dokuments erweitert. `runningTotal`

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$sum` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten. Die folgenden Beispiele zeigen sowohl die Verwendung des Akkumulators (in`$group`) als auch die Verwendung von Fensteroperatoren (in`$setWindowFields`):

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const uri = 'mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false';
  const client = new MongoClient(uri);

  try {
    await client.connect();

    const db = client.db('test');

    // Accumulator usage: sum per group
    const sales = db.collection('sales');
    await sales.insertMany([
      { product: "abc", price: 10, quantity: 2 },
      { product: "abc", price: 10, quantity: 3 },
```

```
      { product: "xyz", price: 20, quantity: 1 },
      { product: "xyz", price: 20, quantity: 5 }
    ]);
    const accumulatorResult = await sales.aggregate([
      { $group: {
          _id: "$product",
          totalSales: { $sum: { $multiply: [ "$price", "$quantity" ] } }
        }}
    ]).toArray();
    console.log('Accumulator result:', accumulatorResult);

    // Window operator usage: running total within each partition
    const dailySales = db.collection('dailySales');
    await dailySales.insertMany([
      { _id: 1, product: "abc", day: 1, amount: 10 },
      { _id: 2, product: "abc", day: 2, amount: 20 },
      { _id: 3, product: "abc", day: 3, amount: 15 },
      { _id: 4, product: "xyz", day: 1, amount: 30 },
      { _id: 5, product: "xyz", day: 2, amount: 25 }
    ]);
    const windowResult = await dailySales.aggregate([
      {
        $setWindowFields: {
          partitionBy: "$product",
          sortBy: { day: 1 },
          output: {
            runningTotal: {
              $sum: "$amount",
              window: { documents: ["unbounded", "current"] }
            }
          }
        }
      }
    ]).toArray();
    console.log('Window result:', windowResult);

  } catch (error) {
    console.error('Error:', error);
  } finally {
    await client.close();
  }
}
```

```
example();
```

Python

```
from pymongo import MongoClient
from pprint import pprint

def example():
    client = None
    try:
        client = MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

        db = client.test

        # Accumulator usage: sum per group
        sales = db.sales
        sales.insert_many([
            { 'product': 'abc', 'price': 10, 'quantity': 2 },
            { 'product': 'abc', 'price': 10, 'quantity': 3 },
            { 'product': 'xyz', 'price': 20, 'quantity': 1 },
            { 'product': 'xyz', 'price': 20, 'quantity': 5 }
        ])
        accumulator_result = list(sales.aggregate([
            { '$group': {
                '_id': '$product',
                'totalSales': { '$sum': { '$multiply': [ '$price', '$quantity' ] } }
            }}
        ]))
        pprint(accumulator_result)

        # Window operator usage: running total within each partition
        daily_sales = db.dailySales
        daily_sales.insert_many([
            { '_id': 1, 'product': 'abc', 'day': 1, 'amount': 10 },
            { '_id': 2, 'product': 'abc', 'day': 2, 'amount': 20 },
            { '_id': 3, 'product': 'abc', 'day': 3, 'amount': 15 },
            { '_id': 4, 'product': 'xyz', 'day': 1, 'amount': 30 },
            { '_id': 5, 'product': 'xyz', 'day': 2, 'amount': 25 }
        ])
        window_result = list(daily_sales.aggregate([
            {
```

```
                '$setWindowFields': {
                    'partitionBy': '$product',
                    'sortBy': { 'day': 1 },
                    'output': {
                        'runningTotal': {
                            '$sum': '$amount',
                            'window': { 'documents': ['unbounded', 'current'] }
                        }
                    }
                }
            }
        ]))
        pprint(window_result)

    except Exception as e:
        print(f"An error occurred: {e}")

    finally:
        if client:
            client.close()

example()
```

$switch

Neu ab Version 4.0.

Wird von Elastic Cluster nicht unterstützt.

Der `$switch` Operator ist ein Operator für bedingte Ausdrücke in Amazon DocumentDB, der es Ihnen ermöglicht, eine Liste von Groß- und Kleinschreibung auszuwerten und den Wert des ersten Falls zurückzugeben, der als wahr ausgewertet wird, oder einen Standardwert, wenn kein Groß-/Kleinschreibung wahr ist.

Parameter

- `branches`: Ein Array von Dokumenten, von denen jedes ein Fallfeld hat, das den auszuwertenden booleschen Ausdruck enthält, und ein Then-Feld, das den Wert enthält, der zurückgegeben werden soll, wenn der Groß-/Kleinschreibung wahr ist.
- `default`: (optional) Der Wert, der zurückgegeben werden soll, wenn keiner der Groß- und Kleinschreibung zutrifft.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$switch` Operator verwendet wird, um die Versandkosten für eine Bestellung anhand der Gesamtsumme der Bestellung zu ermitteln.

Erstellen Sie Beispieldokumente

```
db.orders.insertMany([
  { _id: 1, total: 50 },
  { _id: 2, total: 150 },
  { _id: 3, total: 250 }
]);
```

Beispiel für eine Abfrage

```
db.orders.aggregate([
  {
    $project: {
      _id: 1,
      total: 1,
      shippingCost: {
        $switch: {
          branches: [
            { case: { $lte: ["$total", 100] }, then: 5 },
            { case: { $lte: ["$total", 200] }, then: 10 },
            { case: { $gt: ["$total", 200] }, then: 15 }
          ],
          default: 0
        }
      }
    }
  }
])
```

Ausgabe

```
[
  {
    "_id": 1,
    "total": 50,
    "shippingCost": 5
  },
  {
```

```
    "_id": 2,
    "total": 150,
    "shippingCost": 10
  },
  {
    "_id": 3,
    "total": 250,
    "shippingCost": 15
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$switch` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('orders');

  const result = await collection.aggregate([
    {
      $project: {
        _id: 1,
        total: 1,
        shippingCost: {
          $switch: {
            branches: [
              { case: { $lte: ['$total', 100] }, then: 5 },
              { case: { $lte: ['$total', 200] }, then: 10 },
              { case: { $gt: ['$total', 200] }, then: 15 }
            ],
            default: 0
          }
        }
      }
```

```
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.orders

    result = list(collection.aggregate([
        {
            '$project': {
                '_id': 1,
                'total': 1,
                'shippingCost': {
                    '$switch': {
                        'branches': [
                            { 'case': { '$lte': ['$total', 100] }, 'then': 5 },
                            { 'case': { '$lte': ['$total', 200] }, 'then': 10 },
                            { 'case': { '$gt': ['$total', 200] }, 'then': 15 }
                        ],
                        'default': 0
                    }
                }
            }
        }
    ]))

    print(result)
    client.close()

if __name__ == '__main__':
```

```
    main()
```

$tan

Neu ab Version 8.0.1.

Der `$tan` Operator in Amazon DocumentDB gibt den Tangens eines Werts zurück, der im Bogenmaß gemessen wird.

Der Eingabeausdruck muss in einen numerischen Wert aufgelöst werden. Wenn Ihr Wert in Grad angegeben ist, wenden Sie ihn an, `$degreesToRadians` bevor Sie ihn an übergeben`$tan`.

Der Rückgabetyp ist `double` standardmäßig. Wenn die Eingabe eine 128-Bit-Dezimalzahl ist, ist die Ausgabe ebenfalls eine 128-Bit-Dezimalzahl.

Parameter

- `expression`: Ein Ausdruck, der in eine Zahl im Bogenmaß aufgelöst wird.

Behavior

Null, NaN und +/- Unendlich

Beispiel	Ergebnisse
`{ $tan: NaN }`	`NaN`
`{ $tan: null }`	`null`
`{ $tan: Infinity }` oder `{ $tan: -Infinity }`	Wirft einen Fehler aus.

Wenn die Eingabe erfolgt ist `null` oder das Feld, auf das verwiesen wird, fehlt, wird `$tan` zurückgegeben`null`. Eine Eingabe von `NaN` erzeugt`NaN`. Eine positive oder negative Unendlichkeit verursacht einen Fehler, da die Tangente bei diesen Werten nicht definiert ist.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$tan` Operator verwendet wird, um den Tangens eines Winkels im Bogenmaß zu berechnen.

Erstellen Sie Beispieldokumente

```
db.angles.insertMany([
  { "_id": 1, "angle": 0 },
  { "_id": 2, "angle": 0.7853981633974483 },
  { "_id": 3, "angle": -0.7853981633974483 }
]);
```

Beispiel für eine Abfrage

```
db.angles.aggregate([
  { $project: {
    "tangent": { $tan: "$angle" }
  }}
]);
```

Ausgabe

```
[
  { "_id": 1, "tangent": 0 },
  { "_id": 2, "tangent": 0.9999999999999999 },
  { "_id": 3, "tangent": -0.9999999999999999 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$tan` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
```

```
  const db = client.db('test');
  const collection = db.collection('angles');

  const result = await collection.aggregate([
    { $project: {
      "tangent": { $tan: "$angle" }
    }}
  ]).toArray();

  console.log(result);
  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['angles']

    result = list(collection.aggregate([
        { '$project': {
            'tangent': { '$tan': '$angle' }
        }}
    ]))

    print(result)
    client.close()

if __name__ == "__main__":
    main()
```

$tanh

Neu ab Version 8.0.1.

Der `$tanh` Operator in Amazon DocumentDB gibt den hyperbolischen Tangens eines Werts zurück.

Der Eingabeausdruck muss in einen numerischen Wert aufgelöst werden.

Der Rückgabetyp ist `double` standardmäßig. Wenn die Eingabe eine 128-Bit-Dezimalzahl ist, ist die Ausgabe ebenfalls eine 128-Bit-Dezimalzahl.

Parameter

- `expression`: Ein Ausdruck, der in eine Zahl aufgelöst wird.

Behavior

Null, NaN und +/- Unendlich

Beispiel	Ergebnisse
`{ $tanh: NaN }`	`NaN`
`{ $tanh: null }`	`null`
`{ $tanh: -Infinity }`	`-1`
`{ $tanh: Infinity }`	`1`

Wenn die Eingabe ist `null` oder das Feld, auf das verwiesen wird, fehlt, wird `$tanh` zurückgegeben`null`. Eine Eingabe von `NaN` erzeugt`NaN`. Für positiv unendlich ist das Ergebnis`1`; für negativ unendlich ist das Ergebnis`-1`.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$tanh` Operator verwendet wird, um den hyperbolischen Tangens eines Werts zu berechnen.

Erstellen Sie Beispieldokumente

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 1 },
```

```
  { "_id": 3, "value": -1 }
]);
```

Beispiel für eine Abfrage

```
db.values.aggregate([
  { $project: {
    "result": { $tanh: "$value" }
  }}
]);
```

Ausgabe

```
[
  { "_id": 1, "result": 0 },
  { "_id": 2, "result": 0.7615941559557649 },
  { "_id": 3, "result": -0.7615941559557649 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$tanh` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('values');

  const result = await collection.aggregate([
    { $project: {
      "result": { $tanh: "$value" }
    }}
  ]).toArray();

  console.log(result);
```

```
  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['values']

    result = list(collection.aggregate([
        { '$project': {
            'result': { '$tanh': '$value' }
        }}
    ]))

    print(result)
    client.close()

if __name__ == "__main__":
    main()
```

$top

Neu ab Version 8.0.1.

Verwenden Sie den `$top` Akkumulator in der `$group` Phase, um das Dokument mit dem höchsten Rang pro Gruppe gemäß einer bestimmten Sortierreihenfolge zurückzugeben.

Parameter

- `sortBy`: Ein Dokument, das die Sortierreihenfolge angibt. Wird 1 für aufsteigend oder `-1` für absteigend verwendet.
- `output`: Ein Ausdruck, der die Felder angibt, die aus dem obersten Dokument zurückgegeben werden sollen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$top` Akkumulator verwendet wird, um den Top-Verkauf (höchste Menge) pro Artikel in einer Verkaufskollektion zu ermitteln.

Erstellen Sie Beispieldokumente

```
db.sales.insertMany([
  { item: "abc", quantity: 10, price: 5 },
  { item: "abc", quantity: 5, price: 8 },
  { item: "xyz", quantity: 15, price: 3 },
  { item: "xyz", quantity: 7, price: 6 }
])
```

Beispiel für eine Abfrage

```
db.sales.aggregate([
  { $group: { _id: "$item", topSale: { $top: { sortBy: { quantity: -1 }, output:
 { quantity: "$quantity", price: "$price" } } } } }
])
```

Ausgabe

```
[
  { "_id": "xyz", "topSale": { "quantity": 15, "price": 3 } },
  { "_id": "abc", "topSale": { "quantity": 10, "price": 5 } }
]
```

Beispiel für die Verwendung von Fensteroperatoren (MongoDB Shell)

Neu ab Version 8.0.2.

Der `$top` Operator kann auch als Fensterbediener auf der `$setWindowFields` Bühne verwendet werden. In diesem Zusammenhang gibt er das `output` Dokument zurück, das unter den Dokumenten in jedem Fenster an oberster Stelle steht (entsprechend den Angaben des Operators`sortBy`). Sie geben den Operator unter dem `output` Feld an und definieren optional die Fenstergrenzen mit einem `window` Dokument.

> **Note**
>
> Bei Verwendung als Fensteroperator in `$setWindowFields` `$top` ist er auf 100 MB an Zwischendaten begrenzt. Eine Operation, die dieses Limit überschreitet, gibt einen Fehler zurück.

Erstellen Sie Beispieldokumente

```
db.matchScores.insertMany([
  { _id: 1, player: "Alice", round: 1, score: 20 },
  { _id: 2, player: "Alice", round: 2, score: 35 },
  { _id: 3, player: "Alice", round: 3, score: 28 },
  { _id: 4, player: "Bob", round: 1, score: 15 },
  { _id: 5, player: "Bob", round: 2, score: 40 }
]);
```

Beispiel für eine Abfrage

Das folgende Beispiel partitioniert die Dokumente nach`player`, sortiert jede Partition nach `round` und gibt die Runde mit der höchsten Punktzahl vom Anfang der Partition bis zum aktuellen Dokument zurück.

```
db.matchScores.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$player",
      sortBy: { round: 1 },
      output: {
        bestRoundSoFar: {
          $top: { sortBy: { score: -1 }, output: { round: "$round", score:
 "$score" } },
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "player": "Alice", "round": 1, "score": 20, "bestRoundSoFar": { "round":
 1, "score": 20 } },
  { "_id": 2, "player": "Alice", "round": 2, "score": 35, "bestRoundSoFar": { "round":
 2, "score": 35 } },
  { "_id": 3, "player": "Alice", "round": 3, "score": 28, "bestRoundSoFar": { "round":
 2, "score": 35 } },
  { "_id": 4, "player": "Bob", "round": 1, "score": 15, "bestRoundSoFar": { "round": 1,
 "score": 15 } },
  { "_id": 5, "player": "Bob", "round": 2, "score": 40, "bestRoundSoFar": { "round": 2,
 "score": 40 } }
]
```

Jedes Dokument wird um die Runde mit `bestRoundSoFar` der höchsten Punktzahl innerhalb seiner Partition bis einschließlich des aktuellen Dokuments erweitert.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$top` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten. Die folgenden Beispiele zeigen sowohl die Verwendung des Akkumulators (in`$group`) als auch die Verwendung von Fensteroperatoren (in`$setWindowFields`):

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const uri = 'mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false';
  const client = new MongoClient(uri);

  try {
    await client.connect();

    const db = client.db('test');

    // Accumulator usage: top document per group
    const sales = db.collection('sales');
    await sales.insertMany([
      { item: "abc", quantity: 10, price: 5 },
```

```
      { item: "abc", quantity: 5, price: 8 },
      { item: "xyz", quantity: 15, price: 3 },
      { item: "xyz", quantity: 7, price: 6 }
    ]);
    const accumulatorResult = await sales.aggregate([
      { $group: { _id: "$item", topSale: { $top: { sortBy: { quantity: -1 }, output:
 { quantity: "$quantity", price: "$price" } } } } }
    ]).toArray();
    console.log('Accumulator result:', accumulatorResult);

    // Window operator usage: best round so far within each partition
    const matchScores = db.collection('matchScores');
    await matchScores.insertMany([
      { _id: 1, player: "Alice", round: 1, score: 20 },
      { _id: 2, player: "Alice", round: 2, score: 35 },
      { _id: 3, player: "Alice", round: 3, score: 28 },
      { _id: 4, player: "Bob", round: 1, score: 15 },
      { _id: 5, player: "Bob", round: 2, score: 40 }
    ]);
    const windowResult = await matchScores.aggregate([
      {
        $setWindowFields: {
          partitionBy: "$player",
          sortBy: { round: 1 },
          output: {
            bestRoundSoFar: {
              $top: { sortBy: { score: -1 }, output: { round: "$round", score:
 "$score" } },
              window: { documents: ["unbounded", "current"] }
            }
          }
        }
      }
    ]).toArray();
    console.log('Window result:', windowResult);

  } catch (error) {
    console.error('Error:', error);
  } finally {
    await client.close();
  }
}
```

```
example();
```

Python

```
from pymongo import MongoClient
from pprint import pprint

def example():
    client = None
    try:
        client = MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

        db = client['test']

        # Accumulator usage: top document per group
        sales = db['sales']
        sales.insert_many([
            { 'item': 'abc', 'quantity': 10, 'price': 5 },
            { 'item': 'abc', 'quantity': 5, 'price': 8 },
            { 'item': 'xyz', 'quantity': 15, 'price': 3 },
            { 'item': 'xyz', 'quantity': 7, 'price': 6 }
        ])
        accumulator_result = list(sales.aggregate([
            { '$group': { '_id': '$item', 'topSale': { '$top': { 'sortBy':
 { 'quantity': -1 }, 'output': { 'quantity': '$quantity', 'price':
 '$price' } } } } }
        ]))
        pprint(accumulator_result)

        # Window operator usage: best round so far within each partition
        match_scores = db['matchScores']
        match_scores.insert_many([
            { '_id': 1, 'player': 'Alice', 'round': 1, 'score': 20 },
            { '_id': 2, 'player': 'Alice', 'round': 2, 'score': 35 },
            { '_id': 3, 'player': 'Alice', 'round': 3, 'score': 28 },
            { '_id': 4, 'player': 'Bob', 'round': 1, 'score': 15 },
            { '_id': 5, 'player': 'Bob', 'round': 2, 'score': 40 }
        ])
        window_result = list(match_scores.aggregate([
            {
                '$setWindowFields': {
```

```
                    'partitionBy': '$player',
                    'sortBy': { 'round': 1 },
                    'output': {
                        'bestRoundSoFar': {
                            '$top': { 'sortBy': { 'score': -1 }, 'output':
 { 'round': '$round', 'score': '$score' } },
                            'window': { 'documents': ['unbounded', 'current'] }
                        }
                    }
                }
            }
        ]))
        pprint(window_result)

    except Exception as e:
        print(f"An error occurred: {e}")

    finally:
        if client:
            client.close()

example()
```

$ zu PN

Neu ab Version 8.0.1.

Verwenden Sie den `$topN` Akkumulator in der `$group` Phase, um die obersten N Elemente innerhalb einer Gruppe gemäß einer bestimmten Sortierreihenfolge zurückzugeben. Wenn eine Gruppe weniger als N Elemente enthält, werden alle Elemente in der Gruppe `$topN` zurückgegeben.

Parameter

- n: Eine positive Ganzzahl oder ein Ausdruck, der in eins aufgelöst wird und angibt, wie viele Top-Ergebnisse pro Gruppe zurückgegeben werden sollen.
- `sortBy`: Ein Dokument, das die Sortierreihenfolge angibt. Wird 1 für aufsteigend oder `-1` für absteigend verwendet.
- `output`: Ein Ausdruck, der die Felder angibt, die aus jedem der N wichtigsten Dokumente zurückgegeben werden sollen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$topN` Akkumulator verwendet wird, um die beiden besten Verkäufe (höchste Menge) pro Artikel in einer Verkaufskollektion zu ermitteln.

Erstellen Sie Beispieldokumente

```
db.sales.insertMany([
  { item: "abc", quantity: 10, price: 5 },
  { item: "abc", quantity: 7, price: 8 },
  { item: "abc", quantity: 5, price: 10 },
  { item: "xyz", quantity: 15, price: 3 },
  { item: "xyz", quantity: 9, price: 6 },
  { item: "xyz", quantity: 3, price: 12 }
])
```

Beispiel für eine Abfrage

```
db.sales.aggregate([
  { $group: { _id: "$item", topTwoSales: { $topN: { n: 2, sortBy: { quantity: -1 },
 output: { quantity: "$quantity", price: "$price" } } } } }
])
```

Ausgabe

```
[
  { "_id": "xyz", "topTwoSales": [{ "quantity": 15, "price": 3 }, { "quantity": 9,
 "price": 6 }] },
  { "_id": "abc", "topTwoSales": [{ "quantity": 10, "price": 5 }, { "quantity": 7,
 "price": 8 }] }
]
```

Beispiel für die Verwendung von Fensteroperatoren (MongoDB Shell)

Neu ab Version 8.0.2.

Der `$topN` Operator kann auch als Fensterbediener auf der `$setWindowFields` Bühne verwendet werden. In diesem Zusammenhang gibt er unter den Dokumenten in jedem Fenster ein Array mit den N Dokumenten `output` aus den obersten N Dokumenten zurück (entsprechend den Angaben des Operators`sortBy`). Sie geben den Operator unter dem `output` Feld an und definieren optional die Fenstergrenzen mit einem `window` Dokument.

> **Note**
>
> Bei Verwendung als Fensteroperator in `$setWindowFields` `$topN` ist er auf 100 MB an Zwischendaten begrenzt. Eine Operation, die dieses Limit überschreitet, gibt einen Fehler zurück.

Erstellen Sie Beispieldokumente

```
db.matchScores.insertMany([
  { _id: 1, player: "Alice", round: 1, score: 20 },
  { _id: 2, player: "Alice", round: 2, score: 35 },
  { _id: 3, player: "Alice", round: 3, score: 28 },
  { _id: 4, player: "Bob", round: 1, score: 15 },
  { _id: 5, player: "Bob", round: 2, score: 40 }
]);
```

Beispiel für eine Abfrage

Das folgende Beispiel partitioniert die Dokumente nach`player`, sortiert jede Partition nach `round` und gibt die beiden Runden mit der höchsten Punktzahl vom Anfang der Partition bis zum aktuellen Dokument zurück.

```
db.matchScores.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$player",
      sortBy: { round: 1 },
      output: {
        topTwoRoundsSoFar: {
          $topN: { n: 2, sortBy: { score: -1 }, output: { round: "$round", score:
 "$score" } },
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "player": "Alice", "round": 1, "score": 20, "topTwoRoundsSoFar":
 [{ "round": 1, "score": 20 }] },
  { "_id": 2, "player": "Alice", "round": 2, "score": 35, "topTwoRoundsSoFar":
 [{ "round": 2, "score": 35 }, { "round": 1, "score": 20 }] },
  { "_id": 3, "player": "Alice", "round": 3, "score": 28, "topTwoRoundsSoFar":
 [{ "round": 2, "score": 35 }, { "round": 3, "score": 28 }] },
  { "_id": 4, "player": "Bob", "round": 1, "score": 15, "topTwoRoundsSoFar":
 [{ "round": 1, "score": 15 }] },
  { "_id": 5, "player": "Bob", "round": 2, "score": 40, "topTwoRoundsSoFar":
 [{ "round": 2, "score": 40 }, { "round": 1, "score": 15 }] }
]
```

Jedes Dokument wird um die beiden Runden mit `topTwoRoundsSoFar` der höchsten Punktzahl innerhalb der Partition bis einschließlich des aktuellen Dokuments erweitert. Wenn das Fenster weniger als zwei Dokumente enthält, werden alle zurückgegeben.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$topN` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten. Die folgenden Beispiele zeigen sowohl die Verwendung des Akkumulators (in`$group`) als auch die Verwendung von Fensteroperatoren (in`$setWindowFields`):

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const uri = 'mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false';
  const client = new MongoClient(uri);

  try {
    await client.connect();

    const db = client.db('test');

    // Accumulator usage: top N documents per group
    const sales = db.collection('sales');
    await sales.insertMany([
```

```
      { item: "abc", quantity: 10, price: 5 },
      { item: "abc", quantity: 7, price: 8 },
      { item: "abc", quantity: 5, price: 10 },
      { item: "xyz", quantity: 15, price: 3 },
      { item: "xyz", quantity: 9, price: 6 },
      { item: "xyz", quantity: 3, price: 12 }
    ]);
    const accumulatorResult = await sales.aggregate([
      { $group: { _id: "$item", topTwoSales: { $topN: { n: 2, sortBy: { quantity:
-1 }, output: { quantity: "$quantity", price: "$price" } } } } } }
    ]).toArray();
    console.log('Accumulator result:', accumulatorResult);

    // Window operator usage: top two rounds so far within each partition
    const matchScores = db.collection('matchScores');
    await matchScores.insertMany([
      { _id: 1, player: "Alice", round: 1, score: 20 },
      { _id: 2, player: "Alice", round: 2, score: 35 },
      { _id: 3, player: "Alice", round: 3, score: 28 },
      { _id: 4, player: "Bob", round: 1, score: 15 },
      { _id: 5, player: "Bob", round: 2, score: 40 }
    ]);
    const windowResult = await matchScores.aggregate([
      {
        $setWindowFields: {
          partitionBy: "$player",
          sortBy: { round: 1 },
          output: {
            topTwoRoundsSoFar: {
              $topN: { n: 2, sortBy: { score: -1 }, output: { round: "$round",
score: "$score" } },
              window: { documents: ["unbounded", "current"] }
            }
          }
        }
      }
    ]).toArray();
    console.log('Window result:', windowResult);

  } catch (error) {
    console.error('Error:', error);
  } finally {
    await client.close();
  }
```

```
}

example();
```

Python

```
from pymongo import MongoClient
from pprint import pprint

def example():
    client = None
    try:
        client = MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

        db = client['test']

        # Accumulator usage: top N documents per group
        sales = db['sales']
        sales.insert_many([
            { 'item': 'abc', 'quantity': 10, 'price': 5 },
            { 'item': 'abc', 'quantity': 7, 'price': 8 },
            { 'item': 'abc', 'quantity': 5, 'price': 10 },
            { 'item': 'xyz', 'quantity': 15, 'price': 3 },
            { 'item': 'xyz', 'quantity': 9, 'price': 6 },
            { 'item': 'xyz', 'quantity': 3, 'price': 12 }
        ])
        accumulator_result = list(sales.aggregate([
            { '$group': { '_id': '$item', 'topTwoSales': { '$topN': { 'n': 2,
 'sortBy': { 'quantity': -1 }, 'output': { 'quantity': '$quantity', 'price':
 '$price' } } } } }
        ]))
        pprint(accumulator_result)

        # Window operator usage: top two rounds so far within each partition
        match_scores = db['matchScores']
        match_scores.insert_many([
            { '_id': 1, 'player': 'Alice', 'round': 1, 'score': 20 },
            { '_id': 2, 'player': 'Alice', 'round': 2, 'score': 35 },
            { '_id': 3, 'player': 'Alice', 'round': 3, 'score': 28 },
            { '_id': 4, 'player': 'Bob', 'round': 1, 'score': 15 },
            { '_id': 5, 'player': 'Bob', 'round': 2, 'score': 40 }
```

```
        ])
        window_result = list(match_scores.aggregate([
            {
                '$setWindowFields': {
                    'partitionBy': '$player',
                    'sortBy': { 'round': 1 },
                    'output': {
                        'topTwoRoundsSoFar': {
                            '$topN': { 'n': 2, 'sortBy': { 'score': -1 }, 'output':
 { 'round': '$round', 'score': '$score' } },
                            'window': { 'documents': ['unbounded', 'current'] }
                        }
                    }
                }
            }
        ]))
        pprint(window_result)

    except Exception as e:
        print(f"An error occurred: {e}")

    finally:
        if client:
            client.close()

example()
```

$trunc

Neu ab Version 8.0.1.

Der `$trunc` Operator in Amazon DocumentDB kürzt eine Zahl auf eine angegebene Dezimalstelle und entfernt Ziffern ohne Rundung.

Parameter

- `number`: Ein Ausdruck, der in eine Zahl aufgelöst wird.
- `place`: Optional. Ein Ganzzahlausdruck zwischen -20 und 100, der die Anzahl der Dezimalstellen angibt, auf die gekürzt werden soll. Ein negativer Wert wird links von der Dezimalstelle gekürzt. Der Standardwert ist 0.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$trunc` Operator verwendet wird, um numerische Werte auf eine Dezimalstelle zu kürzen.

Abschneiden mit einem positiven Stellenwert

```
db.measurements.insertMany([
  {_id: 1, value: 3.456},
  {_id: 2, value: 7.891},
  {_id: 3, value: 12.345}
]);
```

Beispiel für eine Abfrage

```
db.measurements.aggregate([
  { $project: { truncated: { $trunc: ["$value", 1] } } }
]);
```

Ausgabe

```
[
  {_id: 1, truncated: 3.4},
  {_id: 2, truncated: 7.8},
  {_id: 3, truncated: 12.3}
]
```

Mit einem negativen Ortswert kürzen

Wenn negativ `place` ist, werden die Ziffern links von der Dezimalstelle durch Nullen `$trunc` ersetzt. Im folgenden Beispiel werden Werte mit der ersten Ziffer links von der Dezimalstelle gekürzt.

```
db.samples.insertMany([
  {_id: 1, value: 19.25},
  {_id: 2, value: 28.73},
  {_id: 3, value: 34.32}
]);
```

```
db.samples.aggregate([
  { $project: { truncated: { $trunc: ["$value", -1] } } }
]);
```

Ausgabe

```
[
  {_id: 1, truncated: 10},
  {_id: 2, truncated: 20},
  {_id: 3, truncated: 30}
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$trunc` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  try {
    await client.connect();
    const db = client.db('test');
    const collection = db.collection('measurements');
    const result = await collection.aggregate([
      { $project: { truncated: { $trunc: ["$value", 1] } } }
    ]).toArray();
    console.log(result);
  } finally {
    await client.close();
  }
}
example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
```

```
    try:
        db = client['test']
        collection = db['measurements']
        result = list(collection.aggregate([
            {'$project': {'truncated': {'$trunc': ['$value', 1]}}}
        ]))
        print(result)
    finally:
        client.close()

example()
```

$ts Inkrement

Neu ab Version 8.0.1.

Der `$tsIncrement` Operator in Amazon DocumentDB gibt die inkrementierende Ordinalzahl eines Timestamp-Werts als lange Ganzzahl zurück.

Parameter

- `expression`: Ein Ausdruck, der in einen Timestamp aufgelöst wird.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$tsIncrement` Operator verwendet wird, um das Ordinalinkrement aus Timestamp-Werten zu extrahieren.

Erstellen Sie Beispieldokumente

```
db.events.insertMany([
  {_id: 1, ts: Timestamp(1678900000, 1)},
  {_id: 2, ts: Timestamp(1678900000, 2)},
  {_id: 3, ts: Timestamp(1678900001, 1)}
]);
```

Beispiel für eine Abfrage

```
db.events.aggregate([
  { $project: { increment: { $tsIncrement: "$ts" } } }
```

```
]);
```

Ausgabe

```
[
  {_id: 1, increment: Long("1")},
  {_id: 2, increment: Long("2")},
  {_id: 3, increment: Long("1")}
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$tsIncrement` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient, Timestamp } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  try {
    await client.connect();
    const db = client.db('test');
    const collection = db.collection('events');
    const result = await collection.aggregate([
      { $project: { increment: { $tsIncrement: "$ts" } } }
    ]).toArray();
    console.log(result);
  } finally {
    await client.close();
  }
}
example();
```

Python

```
from pymongo import MongoClient

def example():
```

```
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    try:
        db = client['test']
        collection = db['events']
        result = list(collection.aggregate([
            {'$project': {'increment': {'$tsIncrement': '$ts'}}}
        ]))
        print(result)
    finally:
        client.close()

example()
```

$ ts Sekunde

Neu ab Version 8.0.1.

Der `$tsSecond` Operator in Amazon DocumentDB gibt den Sekundenanteil eines Timestamp-Werts als lange Ganzzahl zurück (die Sekunden der Unix-Epoche).

Parameter

- `expression`: Ein Ausdruck, der in einen Zeitstempel aufgelöst wird.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$tsSecond` Operator verwendet wird, um den Sekundenanteil aus Timestamp-Werten zu extrahieren.

Erstellen Sie Beispieldokumente

```
db.events.insertMany([
  {_id: 1, ts: Timestamp(1678900000, 1)},
  {_id: 2, ts: Timestamp(1678900000, 2)},
  {_id: 3, ts: Timestamp(1678900001, 1)}
]);
```

Beispiel für eine Abfrage

```
db.events.aggregate([
  { $project: { seconds: { $tsSecond: "$ts" } } }
]);
```

Ausgabe

```
[
  {_id: 1, seconds: Long("1678900000")},
  {_id: 2, seconds: Long("1678900000")},
  {_id: 3, seconds: Long("1678900001")}
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$tsSecond` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient, Timestamp } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  try {
    await client.connect();
    const db = client.db('test');
    const collection = db.collection('events');
    const result = await collection.aggregate([
      { $project: { seconds: { $tsSecond: "$ts" } } }
    ]).toArray();
    console.log(result);
  } finally {
    await client.close();
  }
}
example();
```

Python

```
from pymongo import MongoClient
```

```
def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    try:
        db = client['test']
        collection = db['events']
        result = list(collection.aggregate([
            {'$project': {'seconds': {'$tsSecond': '$ts'}}}
        ]))
        print(result)
    finally:
        client.close()

example()
```

$ zu Bool

Der `$toBool` Operator in Amazon DocumentDB konvertiert einen Ausdruck in einen booleschen Wert.

Parameter

- `expression`: Der Ausdruck, der in einen booleschen Wert umgewandelt werden soll.

Hinweis: Jede Zeichenfolge wird konvertiert in. `true`

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$toBool` Operator verwendet wird, um Gerätestatuswerte aus verschiedenen Datentypen zu normalisieren.

Erstellen Sie Beispieldokumente

```
db.deviceStates.insertMany([
  { _id: 1, deviceId: "sensor-001", status: true },
  { _id: 2, deviceId: "camera-002", status: 1 },
  { _id: 3, deviceId: "thermostat-003", status: "active" },
  { _id: 4, deviceId: "doorlock-004", status: 0 }
]);
```

Beispiel für eine Abfrage

```
db.deviceStates.aggregate([
  {
    $project: {
      _id: 1,
      deviceId: 1,
      isActive: { $toBool: "$status" }
    }
  }
]);
```

Ausgabe

```
[
  { "_id": 1, "deviceId": "sensor-001", "isActive": true },
  { "_id": 2, "deviceId": "camera-002", "isActive": true },
  { "_id": 3, "deviceId": "thermostat-003", "isActive": true },
  { "_id": 4, "deviceId": "doorlock-004", "isActive": false }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$toBool` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('deviceStates');

  const result = await collection.aggregate([
    {
      $project: {
        _id: 1,
        deviceId: 1,
        isActive: { $toBool: '$status' }
```

```
      }
    }
  ]).toArray();

  console.log(result);

  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['deviceStates']

    result = list(collection.aggregate([
        {
            '$project': {
                '_id': 1,
                'deviceId': 1,
                'isActive': { '$toBool': '$status' }
            }
        }
    ]))

    print(result)

    client.close()

if __name__ == '__main__':
    main()
```

$ bis Datum

Neu ab Version 4.0

Der `$toDate` Aggregationsoperator in Amazon DocumentDB wird verwendet, um eine Datums- oder Datums- und Uhrzeitzeichenfolge in einen BSON-Datumstyp zu konvertieren. Dies ist die umgekehrte Operation des Operators. `$dateToString`

Parameter

- `dateString`: Eine Zeichenkettendarstellung eines Datums oder eines Datums und einer Uhrzeit, das in einen BSON-Datumstyp konvertiert werden soll.
- `format`: (optional) Eine Zeichenfolge, die das Format von angibt. `dateString` Falls nicht angegeben, versucht der Operator, das `dateString` in verschiedenen standardmäßigen Datums- und Uhrzeitformaten zu analysieren.
- `timezone`: (optional) Eine Zeichenfolge, die die Zeitzone darstellt, die für die Konvertierung verwendet werden soll. Falls nicht angegeben, wird die lokale Zeitzone verwendet.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$toDate` Operator verwendet wird, um eine Datumszeichenfolge in einen BSON-Datumstyp zu konvertieren.

Erstellen Sie Beispieldokumente

```
db.events.insertMany([
  { _id: 1, eventName: "Mission Start", eventTime: "2023-04-15T10:30:00Z" },
  { _id: 2, eventName: "Checkpoint Reached", eventTime: "2023-04-15T11:15:00Z" },
  { _id: 3, eventName: "Mission End", eventTime: "2023-04-15T12:00:00Z" }
]);
```

Beispiel für eine Abfrage

```
db.events.aggregate([
  {
    $project: {
      eventName: 1,
      eventTimeDate: { $toDate: "$eventTime" }
    }
  }
]);
```

Ausgabe

```
[
  {
    "_id": 1,
    "eventName": "Mission Start",
    "eventTimeDate": ISODate("2023-04-15T10:30:00Z")
  },
  {
    "_id": 2,
    "eventName": "Checkpoint Reached",
    "eventTimeDate": ISODate("2023-04-15T11:15:00Z")
  },
  {
    "_id": 3,
    "eventName": "Mission End",
    "eventTimeDate": ISODate("2023-04-15T12:00:00Z")
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$toDate` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('events');

  const result = await collection.aggregate([
    {
      $project: {
        eventName: 1,
        eventTimeDate: { $toDate: '$eventTime' }
      }
    }
  ]).toArray();
```

```
  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['events']

    result = list(collection.aggregate([
        {
            '$project': {
                'eventName': 1,
                'eventTimeDate': { '$toDate': '$eventTime' }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$ zu Dezimal

Neu ab Version 4.0

Der `$toDecimal` Operator in Amazon DocumentDB wird verwendet, um einen Wert in den Decimal128-Datentyp zu konvertieren. Dies ist nützlich, wenn Sie präzise Dezimalarithmetik durchführen oder große Dezimalwerte verarbeiten müssen, die mit dem Double-Datentyp nicht genau dargestellt werden können.

Parameter

- `expression`: Der Ausdruck, der in den Decimal128-Datentyp konvertiert werden soll.

Beispiel (MongoDB Shell)

Dieses Beispiel zeigt, wie der `$toDecimal` Operator verwendet wird, um einen Zeichenkettenwert in einen Decimal128-Datentyp zu konvertieren.

Erstellen Sie Beispieldokumente

```
db.numbers.insertOne({ _id: 1, value: "3.14" });
db.numbers.insertOne({ _id: 2, value: "2.71" });
```

Beispiel für eine Abfrage

```
db.numbers.aggregate([
  { $project: {
    _id: 1,
    decimalValue: { $toDecimal: "$value" }
  }}
])
```

Ausgabe

```
[
  { "_id" : 1, "decimalValue" : Decimal128("3.14") },
  { "_id" : 2, "decimalValue" : Decimal128("2.71") }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$toDecimal` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
```

```
  const db = client.db('test');
  const collection = db.collection('numbers');

  const result = await collection.aggregate([
    { $project: {
      _id: 1,
      decimalValue: { $toDecimal: "$value" }
    }}
  ]).toArray();

  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.numbers

    result = list(collection.aggregate([
        {'$project': {
            '_id': 1,
            'decimalValue': {'$toDecimal': '$value'}
        }}
    ]))

    print(result)
    client.close()

example()
```

$ bis verdoppeln

Neu ab Version 4.0

Der `$toDouble` Operator in Amazon DocumentDB wird verwendet, um einen Wert in eine 64-Bit-Gleitkommazahl mit doppelter Genauigkeit umzuwandeln. Dies kann nützlich sein, wenn Sie arithmetische Operationen mit Werten durchführen müssen, die ursprünglich nicht in einem numerischen Format vorliegen.

Parameter

`<expression>`: Der Ausdruck, der in einen doppelten Wert umgewandelt werden soll. Dies kann ein beliebiger gültiger Ausdruck sein, der in einen numerischen Wert, eine Zeichenfolge oder einen booleschen Wert aufgelöst wird.

Beispiel (MongoDB Shell)

Dieses Beispiel zeigt, wie der `$toDouble` Operator verwendet wird, um einen Zeichenkettenwert in einen numerischen Wert umzuwandeln, um eine mathematische Berechnung durchzuführen.

Erstellen Sie Beispieldokumente

```
db.numbers.insertMany([
  { _id: 1, value: "10.5" },
  { _id: 2, value: "20.25" },
  { _id: 3, value: "7" }
])
```

Beispiel für eine Abfrage

```
db.numbers.aggregate([
  {
    $project: {
      _id: 1,
      value: 1,
      double_value: { $toDouble: "$value" },
      double_plus_five: { $add: [{ $toDouble: "$value" }, 5] }
    }
  }
])
```

Ausgabe

```
[
```

```
  { "_id" : 1, "value" : "10.5", "double_value" : 10.5, "double_plus_five" : 15.5 },
  { "_id" : 2, "value" : "20.25", "double_value" : 20.25, "double_plus_five" : 25.25 },
  { "_id" : 3, "value" : "7", "double_value" : 7.0, "double_plus_five" : 12.0 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$toDouble` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('numbers');

  const result = await collection.aggregate([
    {
      $project: {
        _id: 1,
        value: 1,
        double_value: { $toDouble: "$value" },
        double_plus_five: { $add: [{ $toDouble: "$value" }, 5] }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
```

```
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['numbers']

    result = list(collection.aggregate([
        {
            '$project': {
                '_id': 1,
                'value': 1,
                'double_value': { '$toDouble': '$value' },
                'double_plus_five': { '$add': [{ '$toDouble': '$value' }, 5] }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$ zu Int

Neu ab Version 4.0

Der `$toInt` Operator in Amazon DocumentDB wird verwendet, um einen Eingabewert in einen Integer-Datentyp zu konvertieren. Dieser Operator ist nützlich, wenn Sie sicherstellen müssen, dass ein Feld oder ein Ausdruck als Ganzzahl dargestellt wird, was für bestimmte Operationen oder Datenverarbeitungsaufgaben wichtig sein kann.

Parameter

- `expression`: Der Ausdruck, der in eine Ganzzahl umgewandelt werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$toInt` Operator verwendet wird, um einen Zeichenkettenwert in eine Ganzzahl umzuwandeln.

Erstellen Sie Beispieldokumente

```
db.numbers.insertMany([
  { "name": "one", "value": "1" },
  { "name": "hundred", "value": "100" }
]);
```

Beispiel für eine Abfrage

```
db.numbers.aggregate([
  { $project: {
    "_id": 0,
    "name": 1,
    "intValue": { $toInt: "$value" }
  }}
]);
```

Ausgabe

```
{ "name": "one", "intValue": 1 }
{ "name": "hundred", "intValue": 100 }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$toInt` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('numbers');

  const result = await collection.aggregate([
    { $project: {
      "_id": 0,
      "name": 1,
      "intValue": { $toInt: "$value" }
    }}
```

```
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['numbers']

    result = list(collection.aggregate([
        { "$project": {
            "_id": 0,
            "name": 1,
            "intValue": { "$toInt": "$value" }
        }}
    ]))

    print(result)
    client.close()

example()
```

$ zu Long

Neu ab Version 4.0

Der `$toLong` Operator in Amazon DocumentDB wird verwendet, um einen Wert in einen 64-Bit-Datentyp vom Typ Integer (Long) zu konvertieren. Dies kann nützlich sein, wenn Sie arithmetische Operationen oder Vergleiche mit numerischen Werten durchführen müssen, die als Zeichenketten oder andere Datentypen gespeichert werden können.

Parameter

- `expression`: Der Ausdruck, der in eine 64-Bit-Ganzzahl konvertiert werden soll.

Beispiel (MongoDB Shell)

Dieses Beispiel zeigt, wie der `$toLong` Operator verwendet wird, um einen Zeichenkettenwert in eine 64-Bit-Ganzzahl zu konvertieren.

Erstellen Sie Beispieldokumente

```
db.numbers.insertMany([
  { _id: 1, value: "42" },
  { _id: 3, value: "9223372036854775807" }
]);
```

Beispiel für eine Abfrage

```
db.numbers.aggregate([
  {
    $project: {
      _id: 1,
      longValue: { $toLong: "$value" }
    }
  }
])
```

Ausgabe

```
[
  { "_id" : 1, "longValue" : 42 },
  { "_id" : 3, "longValue" : 9223372036854775807 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$toLong` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');
```

```
async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const numbers = db.collection('numbers');

  const result = await numbers.aggregate([
    {
      $project: {
        _id: 1,
        longValue: { $toLong: "$value" }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    numbers = db.numbers

    result = list(numbers.aggregate([
        {
            '$project': {
                '_id': 1,
                'longValue': { '$toLong': '$value' }
            }
        }
    ]))
```

```
    print(result)
    client.close()

example()
```

$toLower

Der `$toLower` Operator in Amazon DocumentDB wird verwendet, um eine Zeichenfolge in Kleinbuchstaben umzuwandeln.

Parameter

- `expression`: Der Zeichenkettenausdruck, der in Kleinbuchstaben umgewandelt werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$toLower` Operator verwendet wird, um das `Desk` Feld in Kleinbuchstaben umzuwandeln.

Erstellen Sie Beispieldokumente

```
db.locations.insertMany([
  { "_id": 1, "Desk": "Düsseldorf-BVV-021" },
  { "_id": 2, "Desk": "Munich-HGG-32a" }
]);
```

Beispiel für eine Abfrage

```
db.locations.aggregate([
  { $project: { item: { $toLower: "$Desk" } } }
]);
```

Ausgabe

```
{ "_id" : 1, "item" : "düsseldorf-bvv-021" }
{ "_id" : 2, "item" : "munich-hgg-32a" }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$toLower` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require("mongodb");

async function main() {
  const client = await MongoClient.connect("mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false");
  const db = client.db("test");
  const collection = db.collection("locations");

  const result = await collection.aggregate([
    { $project: { item: { $toLower: "$Desk" } } }
  ]).toArray();

  console.log(result);

  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient("mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false")
    db = client["test"]
    collection = db["locations"]

    result = list(collection.aggregate([
        { "$project": { "item": { "$toLower": "$Desk" } } }
    ]))

    print(result)
```

```
    client.close()

if __name__ == "__main__":
    main()
```

$ bis ObjectId

Neu ab Version 4.0

Der `$toObjectId` Operator in Amazon DocumentDB wird verwendet, um eine Zeichenkettendarstellung eines in einen tatsächlichen ObjectId Datentyp ObjectId zu konvertieren. Dies kann nützlich sein, wenn Sie mit Daten arbeiten, die als Zeichenkettendarstellungen von gespeichert wurden ObjectIds, da Sie damit Operationen ausführen können, für die der ObjectId Datentyp erforderlich ist.

Parameter

- `expression`: Ein Zeichenkettenausdruck, der einen gültigen Wert darstellt ObjectId.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$toObjectId` Operator verwendet wird, um eine Zeichenkettendarstellung von an in den ObjectId Datentyp ObjectId zu konvertieren.

Erstellen Sie Beispieldokumente

```
db.employees.insertMany([
  { _id: 1, empId:"64e5f8886218c620cf0e8f8a", name: "Carol Smith", employeeId:
 "c720a" },
  { _id: 2, empId:"64e5f94e6218c620cf0e8f8c", name: "Bill Taylor", employeeId:
 "c721a" }
]);
```

Beispiel für eine Abfrage

```
db.employees.aggregate([
  { $project: {
    "empIdAsObjectId": {$toObjectId: "$empId"}}
```

```
  }
]);
```

Ausgabe

```
[
  { _id: 1, empIdAsObjectId: ObjectId('64e5f8886218c620cf0e8f8a') },
  { _id: 2, empIdAsObjectId: ObjectId('64e5f94e6218c620cf0e8f8c') }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$toObjectId` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('employees');

  const result = await collection.aggregate([
    { $project: {
      "empIdAsObjectId": {$toObjectId: "$empId"}}
    }
  ]).toArray();

  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient
```

```
def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['employees']

    result = list(collection.aggregate([
      { "$project": {
        "empIdAsObjectId": {"$toObjectId": "$empId"}}
      }
    ]))

    print(result)
    client.close()

example()
```

$ toString

Neu ab Version 4.0

Der `$toString` Operator in Amazon DocumentDB wird verwendet, um einen Wert eines beliebigen Typs (außer Null) in eine Zeichenkettendarstellung zu konvertieren. Dies kann nützlich sein, wenn Sie Zeichenkettenoperationen mit Werten durchführen müssen, die ursprünglich nicht im Zeichenkettenformat vorliegen.

Parameter

- `expression`: Der Ausdruck, der in eine Zeichenfolge konvertiert werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$toString` Operator verwendet wird, um numerische Werte in Zeichenketten umzuwandeln.

Erstellen Sie Beispieldokumente

```
db.numbers.insertMany([
  { "_id": 1, "value": 42 },
  { "_id": 2, "value": 3.14 }
```

```
]);
```

Beispiel für eine Abfrage

```
db.numbers.aggregate([
  { $project: {
    _id: 1,
    valueAsString: { $toString: "$value" }
  }}
]);
```

Ausgabe

```
{ "_id": 1, "valueAsString": "42" }
{ "_id": 2, "valueAsString": "3.14" }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$toString` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('numbers');

  const result = await collection.aggregate([
    { $project: {
      _id: 1,
      valueAsString: { $toString: '$value' }
    }}
  ]).toArray();

  console.log(result);
  client.close();
}
```

```
example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['numbers']

    result = list(collection.aggregate([
        { '$project': {
            '_id': 1,
            'valueAsString': { '$toString': '$value' }
        }}
    ]))

    print(result)
    client.close()

example()
```

$toUpper

Der `$toUpper` Operator in Amazon DocumentDB wird verwendet, um eine Zeichenfolge in Großbuchstaben umzuwandeln.

Parameter

- `expression`: Der Zeichenkettenausdruck, der in Großbuchstaben umgewandelt werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung des `$toUpper` Operators, um das `Desk` Feld in Großbuchstaben umzuwandeln.

Erstellen Sie Beispieldokumente

```
db.locations.insertMany([
  { "_id": 1, "Desk": "düsseldorf-bvv-021" },
  { "_id": 2, "Desk": "munich-hgg-32a" }
]);
```

Beispiel für eine Abfrage

```
db.locations.aggregate([
  { $project: { item: { $toUpper: "$Desk" } } }
]);
```

Ausgabe

```
{ "_id" : 1, "item" : "DüSSELDORF-BVV-021" }
{ "_id" : 2, "item" : "MUNICH-HGG-32A" }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$toUpper` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('locations');

  const result = await collection.aggregate([
    { $project: { item: { $toUpper: '$Desk' } } }
  ]).toArray();

  console.log(result);

  await client.close();
}
```

```
example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['locations']

    result = list(collection.aggregate([
        { '$project': { 'item': { '$toUpper': '$Desk' } } }
    ]))

    print(result)

    client.close()

example()
```

$ zu UUID

Neu ab Version 8.0.1.

Der `$toUUID` Operator in Amazon DocumentDB konvertiert einen Zeichenkettenwert in eine UUID (Binär-Subtyp 4).

Parameter

- `expression`: Ein Ausdruck, der in eine Zeichenfolge im UUID-Format aufgelöst wird (z. B. „12345678-1234-1234-1234-123456789abc“).

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$toUUID` Operator verwendet wird, um Zeichenkettenwerte in das UUID-Format zu konvertieren.

Erstellen Sie Beispieldokumente

```
db.records.insertMany([
  {_id: 1, uuidStr: "550e8400-e29b-41d4-a716-446655440000"},
  {_id: 2, uuidStr: "6ba7b810-9dad-11d1-80b4-00c04fd430c8"}
]);
```

Beispiel für eine Abfrage

```
db.records.aggregate([
  { $project: { uuid: { $toUUID: "$uuidStr" } } }
]);
```

Ausgabe

```
[
  {_id: 1, uuid: UUID("550e8400-e29b-41d4-a716-446655440000")},
  {_id: 2, uuid: UUID("6ba7b810-9dad-11d1-80b4-00c04fd430c8")}
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$toUUID` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  try {
    await client.connect();
    const db = client.db('test');
    const collection = db.collection('records');
    const result = await collection.aggregate([
      { $project: { uuid: { $toUUID: "$uuidStr" } } }
    ]).toArray();
    console.log(result);
  } finally {
    await client.close();
  }
```

```
}
example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    try:
        db = client['test']
        collection = db['records']
        result = list(collection.aggregate([
            {'$project': {'uuid': {'$toUUID': '$uuidStr'}}}
        ]))
        print(result)
    finally:
        client.close()

example()
```

$trim

Neu ab Version 4.0

Der `$trim` Operator in Amazon DocumentDB wird verwendet, um führende and/or Leerzeichen am Ende aus einer Zeichenfolge zu entfernen.

Parameter

- `input`: Der Zeichenkettenausdruck, der gekürzt werden soll.
- `chars`: (optional) Gibt die Zeichen an, die am Anfang und Ende der Eingabe gekürzt werden sollen. Die Standardeinstellung ist Leerzeichen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$trim` Operator verwendet wird, um führende und nachfolgende Leerzeichen aus einer Zeichenfolge zu entfernen.

Erstellen Sie Beispieldokumente

```
db.people.insertMany([
  { "name": "   John Doe   " },
  { "name": "   Bob Johnson   " }
])
```

Beispiel für eine Abfrage

```
db.people.aggregate([
  { $project: {
    "name": { $trim: {input: "$name"}}
  }}
])
```

Ausgabe

```
[
  { "name": "John Doe" },
  { "name": "Bob Johnson" }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$trim` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('people');

  const result = await collection.aggregate([
    { $project: {
      "name": { $trim: {input: "$name" }}
```

```
    }}
  ]).toArray();

  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['people']

    result = list(collection.aggregate([
        {"$project": {
            "name": {"$trim": {"input": "$name"}}
        }}
    ]))

    print(result)
    client.close()

example()
```

$type

Der `$type` Aggregationsoperator gibt den BSON-Datentyp eines angegebenen Felds zurück. Dies ist nützlich, um den Datentyp von Feldwerten bei Aggregationsvorgängen zu identifizieren.

Parameter

- `expression`: Das Feld oder der Ausdruck, dessen Typ zurückgegeben werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$type` Operator verwendet wird, um den Datentyp des Preisfeldes für jedes Produkt zu identifizieren.

Erstellen Sie Beispieldokumente

```
db.inventory.insertMany([
  { _id: 1, item: "Notebook", price: 15.99 },
  { _id: 2, item: "Pen", price: "2.50" },
  { _id: 3, item: "Eraser", price: 1 },
  { _id: 4, item: "Ruler", price: null }
]);
```

Beispiel für eine Abfrage

```
db.inventory.aggregate([
  {
    $project: {
      item: 1,
      price: 1,
      priceType: { $type: "$price" }
    }
  }
]);
```

Ausgabe

```
[
  { _id: 1, item: 'Notebook', price: 15.99, priceType: 'double' },
  { _id: 2, item: 'Pen', price: '2.50', priceType: 'string' },
  { _id: 3, item: 'Eraser', price: 1, priceType: 'int' },
  { _id: 4, item: 'Ruler', price: null, priceType: 'null' }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$type` Aggregationsoperators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('inventory');

  const result = await collection.aggregate([
    {
      $project: {
        item: 1,
        price: 1,
        priceType: { $type: "$price" }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['inventory']

    result = list(collection.aggregate([
        {
            '$project': {
                'item': 1,
                'price': 1,
```

```
                'priceType': { '$type': '$price' }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$unset

Neu ab Version 8.0

Wird von Elastic Cluster nicht unterstützt.

Die `$unset` Aggregationsphase in Amazon DocumentDB ermöglicht es Ihnen, Felder aus Dokumenten zu entfernen.

Parameter

- `expression`: Feldname oder Liste mit mehreren Feldnamen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie das Feld mithilfe der `$unset` Aggregationsphase entfernt wird`price`.

Erstellen Sie Beispieldokumente

```
db.inventory.insertMany([
  { item: "pencil", quantity: 100, price: 0.24},
  { item: "pen", quantity: 204, price: 1.78 }
]);
```

Beispiel für eine Aggregation

```
db.inventory.aggregate([
  {
    $unset: "price"
```

```
  }
])
```

Ausgabe

```
[
  {
    _id: ObjectId('69248951d66dcae121d2950d'),
    item: 'pencil',
    quantity: 100
  },
  {
    _id: ObjectId('69248951d66dcae121d2950e'),
    item: 'pen',
    quantity: 204
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$unset` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const inventory = db.collection('inventory');

  const result = await inventory.aggregate([
      {
        $unset: "price"
      }
  ]).toArray();

  console.log(result);
  client.close();
```

```
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    inventory = db['inventory']

    result = list(inventory.aggregate([
      {
        "$unset": "price"
      }
    ]))

    print(result)
    client.close()

example()
```

$unwind

Der `$unwind` Operator wird verwendet, um ein Array-Feld aus den Eingabedokumenten zu dekonstruieren, um für jedes Element ein Dokument auszugeben. Dies kann nützlich sein, wenn Sie Operationen an den einzelnen Elementen eines Arrays ausführen möchten, z. B. das Filtern, Sortieren oder Transformieren der Daten.

Parameter

- `path`: Der Pfad zu dem Array-Feld, das rückgängig gemacht werden soll.
- `includeArrayIndex`: (optional) Gibt den Namen des neuen Felds an, das den Index des Array-Elements enthalten soll.
- `preserveNullAndEmptyArrays`: (optional) Bestimmt, ob bei der Operation das Originaldokument beibehalten wird, wenn das Array-Feld Null oder ein leeres Array ist.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der $unwind Operator verwendet wird, um ein Array-Feld zu dekonstruieren und weitere Operationen an den einzelnen Elementen durchzuführen.

Erstellen Sie Beispieldokumente

```
db.people.insertMany([
  { _id: 1, name: "jon", hobbies: ["painting", "dancing", "singing"] },
  { _id: 2, name: "jane", hobbies: ["reading", "swimming"] },
  { _id: 3, name: "jack", hobbies: [] }
])
```

Beispiel für eine Abfrage

```
db.people.aggregate([
  { $unwind: "$hobbies" }
])
```

Ausgabe

```
[
  { _id: 1, name: 'jon', hobbies: 'painting' },
  { _id: 1, name: 'jon', hobbies: 'dancing' },
  { _id: 1, name: 'jon', hobbies: 'singing' },
  { _id: 2, name: 'jane', hobbies: 'reading' },
  { _id: 2, name: 'jane', hobbies: 'swimming' }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des $unwind Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
```

```
  const db = client.db('test');
  const collection = db.collection('people');

  const result = await collection.aggregate([
    { $unwind: '$hobbies' }
  ]).toArray();

  console.log(result);
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['people']

    result = list(collection.aggregate([
        { '$unwind': '$hobbies' }
    ]))

    print(result)
    client.close()

example()
```

$VectorSearch

Neu ab Version 8.0

Wird von Elastic Cluster nicht unterstützt.

Der `$vectorSearch` Operator in Amazon DocumentDB ermöglicht Ihnen die Vektorsuche, eine Methode, die beim maschinellen Lernen verwendet wird, um ähnliche Datenpunkte zu finden, indem ihre Vektordarstellungen anhand von Entfernungs- oder Ähnlichkeitsmetriken verglichen

werden. Diese Funktion kombiniert die Flexibilität und die umfassenden Abfragen einer JSON-based Dokumentendatenbank mit der Leistungsfähigkeit der Vektorsuche und ermöglicht es Ihnen, Anwendungsfälle für maschinelles Lernen und generative KI wie semantische Suche, Produktempfehlungen und mehr zu entwickeln.

Parameter

- `<exact>`(optional): Markierung, die angibt, ob die Suche nach dem exakten nächsten Nachbarn (ENN) oder dem ungefähren nächsten Nachbarn (ANN) ausgeführt werden soll. Der Wert kann einer der folgenden sein:
- false — um eine ANN-Suche auszuführen
- true — um die ENN-Suche auszuführen

Wenn es weggelassen oder auf False gesetzt `numCandidates` ist, ist es erforderlich.

```
- `<index>` : Name of the Vector Search index to use.
- `<limit>` : Number of documents to return in the results.
- `<numCandidates>` (optional): This field is required if 'exact' is false or omitted.
 Number of nearest neighbors to use during the search. Value must be less than or equal
 to (<=) 1000. You can't specify a number less than the number of documents to return
 ('limit').
- `<path>` : Indexed vector type field to search.
- `<queryVector>` : Array of numbers that represent the query vector.
```

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$vectorSearch` Operator verwendet werden kann, um ähnliche Produktbeschreibungen auf der Grundlage ihrer Vektordarstellungen zu finden.

Erstellen Sie Beispieldokumente

```
db.products.insertMany([
  {
    _id: 1,
    name: "Product A",
    description: "A high-quality, eco-friendly product for your home.",
    description_vector: [ 0.2, 0.5, 0.8 ]
  },
  {
    _id: 2,
```

```
    name: "Product B",
    description: "An innovative and modern kitchen appliance.",
    description_vector: [0.7, 0.3, 0.9]
  },
  {
    _id: 3,
    name: "Product C",
    description: "A comfortable and stylish piece of furniture.",
    description_vector: [0.1, 0.2, 0.4]
  }
]);
```

Vektor-Suchindex erstellen

```
db.runCommand(
    {
        createIndexes: "products",
        indexes: [{
            key: {
                "description_vector": "vector"
            },
            vectorOptions: {
                type: "hnsw",
                dimensions: 3,
                similarity: "cosine",
                m: 16,
                efConstruction: 64
            },
            name: "description_index"
        }]
    }
);
```

Beispiel für eine Abfrage

```
db.products.aggregate([
  { $vectorSearch: {
      index: "description_index",
      limit: 2,
      numCandidates: 10,
      path: "description_vector",
      queryVector: [0.1, 0.2, 0.3]
    }
```

```
  }
]);
```

Ausgabe

```
[
  {
    "_id": 1,
    "name": "Product A",
    "description": "A high-quality, eco-friendly product for your home.",
    "description_vector": [ 0.2, 0.5, 0.8 ]
  },
  {
    "_id": 3,
    "name": "Product C",
    "description": "A comfortable and stylish piece of furniture.",
    "description_vector": [ 0.1, 0.2, 0.4 ]
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$vectorSearch` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function findSimilarProducts(queryVector) {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('products');

  const result = await collection.aggregate([
    { $vectorSearch: {
        index: "description_index",
        limit: 2,
        numCandidates: 10,
        path: "description_vector",
```

```
        queryVector: queryVector
      }
    }
  ]).toArray();

  console.log(result);
  client.close();
}

findSimilarProducts([0.1, 0.2, 0.3]);
```

Python

```
from pymongo import MongoClient


def find_similar_products(query_vector):
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.products

    result = list(collection.aggregate([
        {
            '$vectorSearch': {
                'index': "description_index",
                'limit': 2,
                'numCandidates': 10,
                'path': "description_vector",
                'queryVector': query_vector
            }
        }
    ]))

    print(result)
    client.close()

find_similar_products([0.1, 0.2, 0.3])
```

$week

Der $week Operator in Amazon DocumentDB gibt die Wochennummer eines Datums (0-53) auf der Grundlage des ISO 8601-Standards zurück. Die Wochennummer wird auf der Grundlage des Jahres und des Wochentags berechnet, wobei Montag der erste Tag der Woche ist.

Parameter

Keine

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der $week Operator verwendet wird, um die Wochennummer eines bestimmten Datums abzurufen.

Erstellen Sie Beispieldokumente

```
db.events.insertMany([
  { _id: 1, date: new Date("2023-01-01") },
  { _id: 2, date: new Date("2023-01-08") },
  { _id: 3, date: new Date("2023-12-31") }
]);
```

Beispiel für eine Abfrage

```
db.events.aggregate([
  { $project: {
    _id: 1,
    week: { $week: "$date" }
  }}
]);
```

Ausgabe

```
[
  { "_id": 1, "week": 1 },
  { "_id": 2, "week": 2 },
  { "_id": 3, "week": 53 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$week` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const events = db.collection('events');

  const result = await events.aggregate([
    { $project: {
      _id: 1,
      week: { $week: "$date" }
    }}
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
  client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
  db = client['test']
  events = db['events']

  result = list(events.aggregate([
      {'$project': {
          '_id': 1,
```

```
            'week': {'$week': '$date'}
        }}
    ]))

    print(result)

    client.close()

example()
```

$year

Der `$year` Operator in Amazon DocumentDB extrahiert die Jahreskomponente aus einem Datum oder Zeitstempel.

Parameter

- `expression`: Der Datums- oder Zeitstempelausdruck, aus dem die Jahreskomponente extrahiert werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$year` Operator verwendet wird, um die Jahreskomponente aus einem Datumsfeld zu extrahieren.

Erstellen Sie Beispieldokumente

```
db.events.insertMany([
  { "_id": 1, "date": ISODate("2023-04-15T00:00:00Z") },
  { "_id": 3, "date": ISODate("2021-12-31T00:00:00Z") }
]);
```

Beispiel für eine Abfrage

```
db.events.aggregate([
  { $project: { year: { $year: "$date" } } }
]);
```

Ausgabe

```
[
  { "_id": 1, "year": 2023 },
  { "_id": 3, "year": 2021 }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$year` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('events');

  const result = await collection.aggregate([
    { $project: { year: { $year: "$date" } } }
  ]).toArray();

  console.log(result);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['events']
```

```
    result = list(collection.aggregate([
        {'$project': {'year': {'$year': '$date'}}}
    ]))

    print(result)

    client.close()

example()
```

$zip

Der `$zip` Operator in Amazon DocumentDB ermöglicht es Ihnen, mehrere Arrays zu einem einzigen Array von Tupeln (Schlüssel-Wert-Paare) zu kombinieren. Dies ist nützlich, wenn Sie neue Dokumente oder Objekte erstellen müssen, indem Sie Daten aus verschiedenen Quellen oder Arrays innerhalb eines Dokuments kombinieren.

Parameter

- `inputs`: Eine Reihe von Ausdrücken, die in Arrays aufgelöst werden. Diese Arrays werden zu einem einzigen Array von Tupeln kombiniert.
- `useLongestLength`: (optional) Fall`true`, hat das Ausgabe-Array die Länge des längsten Eingabe-Arrays und füllt kürzere Arrays mit Werten auf. `null` Fall`false`, hat das Ausgabe-Array die Länge des kürzesten Eingabe-Arrays.
- `defaults`: (optional) Ein Array von Standardwerten, das für die Tupel verwendet werden soll, wenn das entsprechende Eingabearray kürzer als das längste Eingabearray `useLongestLength` ist und es ist`true`.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$zip` Operator verwendet wird, um zwei Arrays zu einem einzigen Array von Tupeln zu kombinieren.

Erstellen Sie Beispieldokumente

```
db.grades.insert([
  {
    "_id": 1,
```

```
    "name": "John",
    "scores": [90, 85, 92],
    "classes": ["Math", "English", "Science"]
  },
  {
    "_id": 2,
    "name": "Jane",
    "scores": [88, 91, 90, 85],
    "classes": ["Math", "English", "Science", "History"]
  }
])
```

Beispiel für eine Abfrage

```
db.grades.aggregate([
  {
    $project: {
      "name": 1,
      "scoredClasses": {
        $zip: {
          inputs: ["$scores", "$classes"],
          useLongestLength: true,
          defaults: [null, null]
        }
      }
    }
  }
])
```

Ausgabe

```
[
  {
    "_id": 1,
    "name": "John",
    "scoredClasses": [
      [90, "Math"],
      [85, "English"],
      [92, "Science"],
      [null, null]
    ]
  },
  {
```

```
    "_id": 2,
    "name": "Jane",
    "scoredClasses": [
      [88, "Math"],
      [91, "English"],
      [90, "Science"],
      [85, "History"]
    ]
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$zip` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('grades');

  const result = await collection.aggregate([
    {
      $project: {
        "name": 1,
        "scoredClasses": {
          $zip: {
            inputs: ["$scores", "$classes"],
            useLongestLength: true,
            defaults: [null, null]
          }
        }
      }
    }
  ]).toArray();

  console.log(result);
```

```
  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['grades']

    result = list(collection.aggregate([
        {
            '$project': {
                'name': 1,
                'scoredClasses': {
                    '$zip': {
                        'inputs': ['$scores', '$classes'],
                        'useLongestLength': True,
                        'defaults': [None, None]
                    }
                }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

Geodaten

Dieser Abschnitt enthält detaillierte Informationen zu Geodatenoperatoren, die von Amazon DocumentDB unterstützt werden.

Themen

- $geometry
- $geoIntersects
- $geoWithin
- $maxDistance
- $minDistance
- $near
- $nearSphere

$geometry

Der `$geometry` Operator in Amazon DocumentDB wird verwendet, um ein GeoJSON-Geometrieobjekt als Teil einer Geodatenabfrage anzugeben. Dieser Operator wird in Verbindung mit anderen Operatoren für Geodatenabfragen wie `$geoWithin` und `$geoIntersects` zur Durchführung räumlicher Abfragen Ihrer Daten verwendet.

In Amazon DocumentDB unterstützt der `$geometry` Operator die folgenden GeoJSON-Geometrietypen:

- Point
- LineString
- Polygon
- MultiPoint
- MultiLineString
- MultiPolygon
- GeometryCollection

Parameter

- `type`: Der Typ des GeoJSON-Geometrieobjekts, z. B.`Point`,`Polygon`, usw.
- `coordinates`: Ein Array von Koordinaten, das die Geometrie darstellt. Die Struktur des Koordinatenarrays hängt vom Geometrietyp ab.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$geometry` Operator verwendet wird, um eine `$geoIntersects` Abfrage in Amazon DocumentDB durchzuführen.

Erstellen Sie Beispieldokumente

```
db.locations.insertMany([
  {
    "_id": 1,
    "name": "Location 1",
    "location": {
      "type": "Point",
      "coordinates": [-73.983253, 40.753941]
    }
  },
  {
    "_id": 2,
    "name": "Location 2",
    "location": {
      "type": "Polygon",
      "coordinates": [[
        [-73.998427, 40.730309],
        [-73.954348, 40.730309],
        [-73.954348, 40.780816],
        [-73.998427, 40.780816],
        [-73.998427, 40.730309]
      ]]
    }
  }
]);
```

Beispiel für eine Abfrage

```
db.locations.find({
  "location": {
    "$geoIntersects": {
      "$geometry": {
        "type": "Polygon",
        "coordinates": [[
          [-73.998, 40.730],
          [-73.954, 40.730],
          [-73.954, 40.781],
```

```
          [-73.998, 40.781],
          [-73.998, 40.730]
        ]]
      }
    }
  }
})
```

Ausgabe

```
[
  {
    "_id": 2,
    "name": "Location 2",
    "location": {
      "type": "Polygon",
      "coordinates": [
        [
          [-73.998427, 40.730309],
          [-73.954348, 40.730309],
          [-73.954348, 40.780816],
          [-73.998427, 40.780816],
          [-73.998427, 40.730309]
        ]
      ]
    }
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$geometry` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
```

```
  const collection = db.collection('locations');

  const query = {
    "location": {
      "$geoIntersects": {
        "$geometry": {
          "type": "Polygon",
          "coordinates": [[
            [-73.998, 40.730],
            [-73.954, 40.730],
            [-73.954, 40.781],
            [-73.998, 40.781],
            [-73.998, 40.730]
          ]]
        }
      }
    }
  };

  const result = await collection.find(query).toArray();
  console.log(result);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['locations']

    query = {
        "location": {
            "$geoIntersects": {
                "$geometry": {
                    "type": "Polygon",
```

```
                    "coordinates": [[
                        [-73.998, 40.730],
                        [-73.954, 40.730],
                        [-73.954, 40.781],
                        [-73.998, 40.781],
                        [-73.998, 40.730]
                    ]]
                }
            }
        }
    }

    result = list(collection.find(query))
    print(result)

    client.close()

example()
```

$geoIntersects

Der `$geoIntersects` Operator in Amazon DocumentDB wird verwendet, um Dokumente zu finden, deren Geodaten sich mit einem bestimmten GeoJSON-Objekt überschneiden. Dieser Operator ist nützlich für Anwendungen, bei denen Dokumente anhand ihrer räumlichen Beziehung zu einer bestimmten geographischen Form, wie z. B. einem Polygon oder Multipolygon, identifiziert werden müssen.

Parameter

- `$geometry`: Ein GeoJSON-Objekt, das die Form darstellt, bei der nach Überschneidungen gesucht werden soll. Die unterstützten GeoJSON-Objekttypen sind`Point`, `LineStringPolygon`, und`MultiPolygon`.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$geoIntersects` Operator verwendet wird, um den Namen des Bundesstaates für einen bestimmten Satz von Koordinaten in Amazon DocumentDB zu finden.

Erstellen Sie Beispieldokumente

```
db.states.insertMany([
```

```
  {
    "name": "New York",
    "loc": {
      "type": "Polygon",
      "coordinates": [[
        [-74.25909423828125, 40.47556838210948],
        [-73.70819091796875, 40.47556838210948],
        [-73.70819091796875, 41.31342607582222],
        [-74.25909423828125, 41.31342607582222],
        [-74.25909423828125, 40.47556838210948]
      ]]
    }
  },
  {
    "name": "California",
    "loc": {
      "type": "Polygon",
      "coordinates": [[
        [-124.4091796875, 32.56456771381587],
        [-114.5458984375, 32.56456771381587],
        [-114.5458984375, 42.00964153424558],
        [-124.4091796875, 42.00964153424558],
        [-124.4091796875, 32.56456771381587]
      ]]
    }
  }
]);
```

Beispiel für eine Abfrage

```
var location = [-73.965355, 40.782865];

db.states.find({
  "loc": {
    "$geoIntersects": {
      "$geometry": {
        "type": "Point",
        "coordinates": location
      }
    }
  }
}, {
  "name": 1
```

```
});
```

Ausgabe

```
{ "_id" : ObjectId("536b0a143004b15885c91a2c"), "name" : "New York" }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$geoIntersects` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function findStateByGeoIntersects(longitude, latitude) {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('states');

  const query = {
    loc: {
      $geoIntersects: {
        $geometry: {
          type: 'Point',
          coordinates: [longitude, latitude]
        }
      }
    }
  };

  const projection = {
    _id: 0,
    name: 1
  };

  const document = await collection.findOne(query, { projection });

  await client.close();

  if (document) {
```

```
    return document.name;
  } else {
    throw new Error('The geo location you entered was not found in the United
 States!');
  }
}
```

Python

```
from pymongo import MongoClient

def find_state_by_geointersects(longitude, latitude):
    try:
        client = MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
        db = client.test
        collection_states = db.states

        query_geointersects = {
            "loc": {
                "$geoIntersects": {
                    "$geometry": {
                        "type": "Point",
                        "coordinates": [longitude, latitude]
                    }
                }
            }
        }

        document = collection_states.find_one(query_geointersects,
                                              projection={
                                                  "_id": 0,
                                                  "name": 1
                                              })
        if document is not None:
            state_name = document['name']
            return state_name
        else:
            raise Exception("The geo location you entered was not found in the
 United States!")
    except Exception as e:
        print('Exception in geoIntersects: {}'.format(e))
```

```
        raise
    finally:
        if client is not None:
            client.close()
```

$geoWithin

Der `$geoWithin` Operator in Amazon DocumentDB wird verwendet, um Dokumente zu finden, deren Standortdaten (dargestellt als GeoJSON-Objekte) vollständig in einer bestimmten Form enthalten sind, z. B. einem Polygon oder Multipolygon. Dies ist nützlich für die Abfrage von Objekten, die sich in einer bestimmten geografischen Region befinden.

Parameter

- `$geometry`: Ein GeoJSON-Objekt, das die Form darstellt, nach der abgefragt werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$geoWithin` Operator verwendet werden kann, um alle Flughäfen im Bundesstaat New York zu finden.

Erstellen Sie Beispieldokumente

```
// Insert state document
db.states.insert({
    "name": "New York",
    "loc": {
        "type": "Polygon",
        "coordinates": [[
            [-79.76278, 45.0],
            [-73.94, 45.0],
            [-73.94, 40.5],
            [-79.76278, 40.5],
            [-79.76278, 45.0]
        ]]
    }
});

// Insert airport documents
db.airports.insert([
    {
```

```
        "name": "John F. Kennedy International Airport",
        "type": "airport",
        "code": "JFK",
        "loc": {
            "type": "Point",
            "coordinates": [-73.7781, 40.6413]
        }
    },
    {
        "name": "LaGuardia Airport",
        "type": "airport",
        "code": "LGA",
        "loc": {
            "type": "Point",
            "coordinates": [-73.8772, 40.7769]
        }
    },
    {
        "name": "Buffalo Niagara International Airport",
        "type": "airport",
        "code": "BUF",
        "loc": {
            "type": "Point",
            "coordinates": [-78.7322, 42.9403]
        }
    }
]);
```

Beispiel für eine Abfrage

```
var state = db.states.findOne({"name": "New York"});

db.airports.find({
    "loc": {
        "$geoWithin": {
            "$geometry": state.loc
        }
    }
}, {
    "name": 1,
    "type": 1,
    "code": 1,
    "_id": 0
```

```
});
```

Ausgabe

```
[
  {
    "name": "John F. Kennedy International Airport",
    "type": "airport",
    "code": "JFK"
  },
  {
    "name": "LaGuardia Airport",
    "type": "airport",
    "code": "LGA"
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$geoWithin` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function findAirportsWithinState(stateName) {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');

  const stateDoc = await db.collection('states').findOne({ name: stateName },
 { projection: { _id: 0, loc: 1 } });
  const airportDocs = await db.collection('airports').find({
    loc: {
      $geoWithin: {
        $geometry: stateDoc.loc
      }
    }
  }, { projection: { name: 1, type: 1, code: 1, _id: 0 } }).toArray();

  console.log(airportDocs);
```

```
  await client.close();
}

findAirportsWithinState('New York');
```

Python

```
from pymongo import MongoClient

def find_airports_within_state(state_name):
    try:
        client = MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
        db = client['test']
        state_doc = db.states.find_one({'name': state_name}, {'_id': 0, 'loc': 1})
        airport_docs = db.airports.find({
            'loc': {
                '$geoWithin': {
                    '$geometry': state_doc['loc']
                }
            }
        }, {'name': 1, 'type': 1, 'code': 1, '_id': 0})

        return list(airport_docs)
    except Exception as e:
        print(f'Error: {e}')
    finally:
        client.close()

airports = find_airports_within_state('New York')
print(airports)
```

$maxDistance

Der `$maxDistance` Operator in Amazon DocumentDB wird verwendet, um die maximale Entfernung (in Metern) von einem GeoJSON-Punkt anzugeben, innerhalb derer sich Dokumente befinden müssen, um in die Abfrageergebnisse aufgenommen zu werden. Dieser Operator wird in Verbindung mit dem `$nearSphere` Operator verwendet, um Geodatenabfragen durchzuführen.

Parameter

- `$maxDistance`: Die maximale Entfernung (in Metern) vom Referenzpunkt, innerhalb derer sich Dokumente befinden müssen, um in die Abfrageergebnisse aufgenommen zu werden.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$maxDistance` Operator in Amazon DocumentDB verwendet wird, um alle Hauptstädte der Bundesstaaten im Umkreis von 100 Kilometern von Boston zu finden.

Erstellen Sie Beispieldokumente

```
db.capitals.insert([
  { state: "Massachusetts", city: "Boston", location: { type: "Point", coordinates:
 [-71.0589, 42.3601] } },
  { state: "Rhode Island", city: "Providence", location: { type: "Point", coordinates:
 [-71.4128, 41.8239] } },
  { state: "New Hampshire", city: "Concord", location: { type: "Point", coordinates:
 [-71.5383, 43.2067] } },
  { state: "Vermont", city: "Montpelier", location: { type: "Point", coordinates:
 [-72.5751, 44.2604] } }
]);
```

Beispiel für eine Abfrage

```
db.capitals.find(
  {
    location: {
      $nearSphere: {
        $geometry: { type: "Point", coordinates: [-71.0589, 42.3601] },
        $maxDistance: 100000
      }
    }
  },
  { state: 1, city: 1, _id: 0 }
);
```

Ausgabe

```
[
  { "state": "Rhode Island", "city": "Providence" },
  { "state": "New Hampshire", "city": "Concord" }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$maxDistance` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function findCapitalsNearBoston() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const capitals = db.collection('capitals');

  const result = await capitals.find({
    location: {
      $nearSphere: {
        $geometry: { type: "Point", coordinates: [-71.0589, 42.3601] },
        $maxDistance: 100000
      }
    }
  }, {
    projection: { state: 1, city: 1, _id: 0 }
  }).toArray();

  console.log(result);
  await client.close();
}

findCapitalsNearBoston();
```

Python

```
from pymongo import MongoClient

def find_capitals_near_boston():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    capitals = db.capitals
```

```
    result = list(capitals.find(
        {
            "location": {
                "$nearSphere": {
                    "$geometry": { "type": "Point", "coordinates": [-71.0589,
 42.3601] },
                    "$maxDistance": 100000
                }
            }
        },
        {"state": 1, "city": 1, "_id": 0}
    ))

    print(result)
    client.close()

find_capitals_near_boston()
```

$minDistance

`$minDistance`ist ein Suchoperator, der in Verbindung mit `$nearSphere` oder verwendet wird, `$geoNear` um Dokumente zu filtern, die sich mindestens im angegebenen Mindestabstand vom Mittelpunkt befinden. Dieser Operator wird in Amazon DocumentDB unterstützt und funktioniert ähnlich wie sein Gegenstück in MongoDB.

Parameter

- `$minDistance`: Der Mindestabstand (in Metern) vom Mittelpunkt, um Dokumente in die Ergebnisse einzubeziehen.

Beispiel (MongoDB Shell)

In diesem Beispiel finden wir alle Restaurants in einem Umkreis von 2 Kilometern um einen bestimmten Standort in Seattle, Washington.

Erstellen Sie Beispieldokumente

```
db.usarestaurants.insertMany([
  {
    "state": "Washington",
```

```
    "city": "Seattle",
    "name": "Noodle House",
    "rating": 4.8,
    "location": {
      "type": "Point",
      "coordinates": [-122.3517, 47.6159]
    }
  },
  {
    "state": "Washington",
    "city": "Seattle",
    "name": "Pike Place Grill",
    "rating": 4.5,
    "location": {
      "type": "Point",
      "coordinates": [-122.3412, 47.6102]
    }
  },
  {
    "state": "Washington",
    "city": "Bellevue",
    "name": "The Burger Joint",
    "rating": 4.2,
    "location": {
      "type": "Point",
      "coordinates": [-122.2007, 47.6105]
    }
  }
]);
```

Beispiel für eine Abfrage

```
db.usarestaurants.find({
  "location": {
    "$nearSphere": {
      "$geometry": {
        "type": "Point",
        "coordinates": [-122.3516, 47.6156]
      },
      "$minDistance": 1,
      "$maxDistance": 2000
    }
  }
```

```
}, {
  "name": 1
});
```

Ausgabe

```
{ "_id" : ObjectId("611f3da985009a81ad38e74b"), "name" : "Noodle House" }
{ "_id" : ObjectId("611f3da985009a81ad38e74c"), "name" : "Pike Place Grill" }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$minDistance` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function findRestaurantsNearby() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('usarestaurants');

  const result = await collection.find({
    "location": {
      "$nearSphere": {
        "$geometry": {
          "type": "Point",
          "coordinates": [-122.3516, 47.6156]
        },
        "$minDistance": 1,
        "$maxDistance": 2000
      }
    }
  }, {
    "projection": { "name": 1 }
  }).toArray();

  console.log(result);
  client.close();
}
```

```
findRestaurantsNearby();
```

Python

```
from pymongo import MongoClient

def find_restaurants_nearby():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.usarestaurants

    result = list(collection.find({
        "location": {
            "$nearSphere": {
                "$geometry": {
                    "type": "Point",
                    "coordinates": [-122.3516, 47.6156]
                },
                "$minDistance": 1,
                "$maxDistance": 2000
            }
        }
    }, {
        "projection": {"name": 1}
    }))

    print(result)
    client.close()

find_restaurants_nearby()
```

$near

Der `$near` Operator in Amazon DocumentDB wird verwendet, um Dokumente zu finden, die sich geografisch in der Nähe eines bestimmten Punktes befinden. Er gibt Dokumente zurück, die nach Entfernung sortiert sind, wobei die Dokumente, die sich am nächsten befinden, zuerst angezeigt werden. Dieser Operator erfordert einen 2dSphere-Geodatenindex und ist nützlich für Näherungsabfragen von Standortdaten.

Parameter

- `$geometry`: Ein GeoJSON Point-Objekt, das den Mittelpunkt für die Near-Abfrage definiert.
- `$maxDistance`: (optional) Die maximale Entfernung in Metern vom angegebenen Punkt, die ein Dokument haben darf, um der Abfrage zu entsprechen.
- `$minDistance`: (optional) Die Mindestentfernung in Metern vom angegebenen Punkt, die ein Dokument haben darf, um der Abfrage zu entsprechen.

Anforderungen an den Index

- `2dsphere index`: Erforderlich für Geodatenabfragen zu GeoJSON-Punktdaten.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie Sie den `$near` Operator verwenden können, um die Restaurants zu finden, die einem bestimmten Ort in Seattle, Washington, am nächsten liegen.

Erstellen Sie Beispieldokumente

```
db.usarestaurants.insert([
  {
    "name": "Noodle House",
    "city": "Seattle",
    "state": "Washington",
    "rating": 4.8,
    "location": { "type": "Point", "coordinates": [-122.3517, 47.6159] }
  },
  {
    "name": "Pike Place Grill",
    "city": "Seattle",
    "state": "Washington",
    "rating": 4.2,
    "location": { "type": "Point", "coordinates": [-122.3403, 47.6062] }
  },
  {
    "name": "Lola",
    "city": "Seattle",
    "state": "Washington",
    "rating": 4.5,
    "location": { "type": "Point", "coordinates": [-122.3407, 47.6107] }
  }
```

```
]);
```

Erstellen Sie einen 2dSphere-Index

```
db.usarestaurants.createIndex({ "location": "2dsphere" });
```

Abfragebeispiel mit GeoJSON Point

```
db.usarestaurants.find({
  location: {
    $near: {
      $geometry: {
        type: "Point",
        coordinates: [-122.3516, 47.6156]
      },
      $maxDistance: 100,
      $minDistance: 10
    }
  }
});
```

Ausgabe

```
{
  "_id" : ObjectId("69031ec9ea1c2922a1ce5f4a"),
  "name" : "Noodle House",
  "city" : "Seattle",
  "state" : "Washington",
  "rating" : 4.8,
  "location" : {
    "type" : "Point",
    "coordinates" : [ -122.3517, 47.6159 ]
  }
}
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$near` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function findNearbyRestaurants() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const restaurants = db.collection('usarestaurants');

  // Create 2dsphere index
  await restaurants.createIndex({ "location": "2dsphere" });

  const result = await restaurants.find({
    location: {
      $near: {
        $geometry: {
          type: "Point",
          coordinates: [-122.3516, 47.6156]
        },
        $maxDistance: 100,
        $minDistance: 10
      }
    }
  }).toArray();

  console.log(result);

  client.close();
}

findNearbyRestaurants();
```

Python

```
from pymongo import MongoClient

def find_nearby_restaurants():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
```

```
    restaurants = db['usarestaurants']

    # Create 2dsphere index
    restaurants.create_index([("location", "2dsphere")])

    result = list(restaurants.find({
        'location': {
            '$near': {
                '$geometry': {
                    'type': 'Point',
                    'coordinates': [-122.3516, 47.6156]
                },
                '$maxDistance': 100,
                '$minDistance': 10
            }
        }
    }))

    print(result)

    client.close()

find_nearby_restaurants()
```

$nearSphere

Der `$nearSphere` Operator in Amazon DocumentDB wird verwendet, um Dokumente zu finden, die sich innerhalb einer bestimmten Entfernung von einem Geodatenpunkt befinden. Dieser Operator ist besonders nützlich für geografische Abfragen, z. B. für die Suche nach allen Restaurants in einem bestimmten Umkreis eines bestimmten Standorts.

Parameter

- `$geometry`: Ein GeoJSON-Objekt, das den Referenzpunkt darstellt. Muss ein `Point` Objekt mit `coordinates` Feldern `type` und sein.
- `$minDistance`: (optional) Der Mindestabstand (in Metern) vom Referenzpunkt, den Dokumente haben müssen.
- `$maxDistance`: (optional) Der maximale Abstand (in Metern) vom Referenzpunkt, den Dokumente haben müssen.

Beispiel (MongoDB Shell)

In diesem Beispiel finden wir alle Restaurants im Umkreis von 2 Kilometern (2000 Metern) von einem bestimmten Ort in Seattle, WA.

Erstellen Sie Beispieldokumente

```
db.usarestaurants.insert([
  {
    name: "Noodle House",
    location: { type: "Point", coordinates: [-122.3516, 47.6156] }
  },
  {
    name: "Pike Place Grill",
    location: { type: "Point", coordinates: [-122.3403, 47.6101] }
  },
  {
    name: "Seattle Coffee Co.",
    location: { type: "Point", coordinates: [-122.3339, 47.6062] }
  }
]);
```

Beispiel für eine Abfrage

```
db.usarestaurants.find({
  location: {
    $nearSphere: {
      $geometry: {
        type: "Point",
        coordinates: [-122.3516, 47.6156]
      },
      $minDistance: 1,
      $maxDistance: 2000
    }
  }
}, {
  name: 1
});
```

Ausgabe

```
{ "_id" : ObjectId("611f3da985009a81ad38e74b"), "name" : "Noodle House" }
```

```
{ "_id" : ObjectId("611f3da985009a81ad38e74c"), "name" : "Pike Place Grill" }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$nearSphere` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function findNearbyRestaurants() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const restaurants = db.collection('usarestaurants');

  const result = await restaurants.find({
    location: {
      $nearSphere: {
        $geometry: {
          type: "Point",
          coordinates: [-122.3516, 47.6156]
        },
        $minDistance: 1,
        $maxDistance: 2000
      }
    }
  }, {
    projection: { name: 1 }
  }).toArray();

  console.log(result);
  client.close();
}

findNearbyRestaurants();
```

Python

```
from pymongo import MongoClient
```

```
def find_nearby_restaurants():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    restaurants = db.usarestaurants

    result = list(restaurants.find({
        'location': {
            '$nearSphere': {
                '$geometry': {
                    'type': 'Point',
                    'coordinates': [-122.3516, 47.6156]
                },
                '$minDistance': 1,
                '$maxDistance': 2000
            }
        }
    }, {
        'name': 1
    }))

    print(result)
    client.close()

find_nearby_restaurants()
```

Abfrage- und Projektionsoperatoren

Dieser Abschnitt enthält detaillierte Informationen zu Abfrage- und Projektionsoperatoren, die von Amazon DocumentDB unterstützt werden.

Themen

- $
- $all
- $and
- $ Bits AllClear
- $ Bits AllSet
- $ Bits AnyClear

- $ Bits AnySet
- $comment
- $elemMatch
- $eq
- $exists
- $expr
- $gt
- $gte
- $in
- $jsonSchema
- $lt
- $mod
- $lte
- $meta
- $ne
- $nin
- $nor
- $not
- $or
- $regex
- $ Beispielrate
- $slice
- $size
- $text
- $type

$

Der $ Projektionsoperator begrenzt den Inhalt eines Array-Felds so, dass nur das erste Element zurückgegeben wird, das der Abfragebedingung entspricht. Er wird verwendet, um ein einzelnes passendes Array-Element zu projizieren.

Parameter

- `field.$`: Das Array-Feld mit dem Positionsoperator zur Projektion des ersten passenden Elements.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der $ Projektionsoperator verwendet wird, um nur das passende Array-Element zurückzugeben.

Erstellen Sie Beispieldokumente

```
db.students.insertMany([
  { _id: 1, name: "Alice", grades: [85, 92, 78, 95] },
  { _id: 2, name: "Bob", grades: [70, 88, 92, 65] },
  { _id: 3, name: "Charlie", grades: [95, 89, 91, 88] }
]);
```

Beispiel für eine Abfrage

```
db.students.find(
  { grades: { $gte: 90 } },
  { name: 1, "grades.$": 1 }
);
```

Ausgabe

```
{ "_id" : 1, "name" : "Alice", "grades" : [ 92 ] }
{ "_id" : 2, "name" : "Bob", "grades" : [ 92 ] }
{ "_id" : 3, "name" : "Charlie", "grades" : [ 95 ] }
```

In diesem Beispiel wird für jeden Schüler nur die erste Note zurückgegeben, die größer oder gleich 90 ist.

Codebeispiele

Um ein Codebeispiel für die Verwendung des $ Projektionsoperators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('students');

  const result = await collection.find(
    { grades: { $gte: 90 } },
    { projection: { name: 1, "grades.$": 1 } }
  ).toArray();

  console.log(JSON.stringify(result, null, 2));
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['students']

    result = list(collection.find(
        {'grades': {'$gte': 90}},
        {'name': 1, 'grades.$': 1}
    ))

    print(result)
    client.close()

example()
```

$all

Der `$all` Operator in Amazon DocumentDB wird verwendet, um Dokumente abzugleichen, bei denen der Wert eines Felds ein Array ist und alle angegebenen Elemente enthält, unabhängig von der Reihenfolge der Elemente im Array.

Parameter

- `field`: Der Name des zu prüfenden Felds.
- `[value1, value2, ...]`: Die Liste der Werte, die im Array abgeglichen werden sollen.

`$elemMatch`Innerhalb eines **`$all`** Ausdrucks verwenden

Informationen Verwendung von $elemMatch innerhalb eines $all-Ausdrucks zu Einschränkungen in Bezug auf die Verwendung des `$elemMatch` Operators innerhalb eines `$all` Ausdrucks finden Sie unter.

Dollar ($) in Feldnamen

Informationen Dollar ($) und Punkt (.) in Feldnamen zu Einschränkungen beim Abfragen von Feldern mit $ Präfix `$all` in verschachtelten Objekten finden Sie unter.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung des `$all` Operators zum Abrufen von Dokumenten, bei denen das Feld „Colors“ ein Array ist, das sowohl „Red“ als auch „Blue“ enthält.

Erstellen Sie Beispieldokumente

```
db.example.insertMany([
  { "Item": "Pen", "Colors": ["Red", "Blue", "Green"] },
  { "Item": "Notebook", "Colors": ["Blue", "White"] },
  { "Item": "Poster Paint", "Colors": ["Red", "Yellow", "White"] }
])
```

Beispiel für eine Abfrage

```
db.example.find({ "Colors": { $all: ["Red", "Blue"] } }).pretty()
```

Ausgabe

```
{
  "_id" : ObjectId("6137d6c5b3a1d35e0b6ee6ad"),
  "Item" : "Pen",
  "Colors" : [
          "Red",
          "Blue",
          "Green"
  ]
}
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$all` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('example');

  const result = await collection.find({ "Colors": { $all: ["Red",
 "Blue"] } }).toArray();
  console.log(result);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient
```

```
def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['example']

    result = list(collection.find({ "Colors": { "$all": ["Red", "Blue"] } }))
    print(result)

    client.close()

example()
```

$and

Der `$and` Operator in Amazon DocumentDB wird verwendet, um mehrere Ausdrücke zu kombinieren und sie als eine einzige Bedingung auszuwerten. Er wird zurückgegeben`true`, wenn alle angegebenen Ausdrücke als ausgewertet werden`true`, und `false` andernfalls. Dieser Operator ist nützlich, um mehrere Kriterien auf eine Abfrage anzuwenden.

Parameter

- `expression1`: Ein erforderlicher Ausdruck, der einen booleschen Wert ergibt.
- `expression2`: Ein erforderlicher Ausdruck, der einen booleschen Wert ergibt.
- `...`: Zusätzliche erforderliche Ausdrücke, die zu booleschen Werten ausgewertet werden.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$and` Operator verwendet wird, um alle Dokumente in der Sammlung „Benutzer“ zu finden, bei denen das Feld „Alter“ größer als 18 und das Feld „Status“ „aktiv“ ist.

Erstellen Sie Beispieldokumente

```
db.users.insertMany([
  { name: "John", age: 25, status: "active" },
  { name: "Jane", age: 17, status: "active" },
```

```
  { name: "Bob", age: 30, status: "inactive" },
  { name: "Alice", age: 22, status: "active" }
]);
```

Beispiel für eine Abfrage

```
db.users.find({
  $and: [
    { age: { $gt: 18 } },
    { status: "active" }
  ]
});
```

Ausgabe

```
[
  { "_id" : ObjectId("614e3c4b63f5892e7c4e2345"), "name" : "John", "age" : 25,
 "status" : "active" },
  { "_id" : ObjectId("614e3c4b63f5892e7c4e2347"), "name" : "Alice", "age" : 22,
 "status" : "active" }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des $and Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function findActiveUsersOlderThan18() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const users = db.collection('users');

  const activeUsersOlderThan18 = await users.find({
    $and: [
      { age: { $gt: 18 } },
```

```
      { status: 'active' }
    ]
  }).toArray();

  console.log(activeUsersOlderThan18);
  await client.close();
}

findActiveUsersOlderThan18();
```

Python

```
from pymongo import MongoClient

def find_active_users_older_than_18():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    users = db['users']

    active_users_older_than_18 = list(users.find({
        '$and': [
            {'age': {'$gt': 18}},
            {'status': 'active'}
        ]
    }))

    print(active_users_older_than_18)
    client.close()

find_active_users_older_than_18()
```

$ Bits AllClear

Der `$bitsAllClear` Operator in Amazon DocumentDB wird verwendet, um die Dokumente abzugleichen, bei denen alle angegebenen Bits in einem Feld gelöscht (auf 0 gesetzt) wurden. Dieser Operator kann nützlich sein, um bitweise Operationen mit gespeicherten Daten durchzuführen.

Parameter

- `field`: Das Feld, in dem geprüft werden soll, ob die angegebenen Bits gelöscht wurden.

- `value`: Die numerische Bitmaske, die angibt, welche Bits geprüft werden sollen, oder eine Liste von Bitpositionen, die überprüft werden sollen. Eine numerische Bitmaske kann binär (0b...), dezimal, hexadezimal (0x...), oktal (0o...) oder binär () sein. BinData In einer Liste von Bitpositionen ist die Position des niedrigstwertigen Bits 0.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung des `$bitsAllClear` Operators in Amazon DocumentDB.

Erstellen Sie Beispieldokumente

```
db.collection.insertMany([
  { _id: 1, bits: 0b1010 },
  { _id: 2, bits: 0b1100 },
  { _id: 3, bits: 0b0101 }
]);
```

Beispiel für eine Abfrage

```
db.collection.find({
  bits: { $bitsAllClear: 0b0011 }
})
```

Ausgabe

```
{ "_id" : 2, "bits" : 12 }
```

Die Abfrage prüft, ob alle durch die Bitmaske angegebenen Bits `0b0011` (die zwei niedrigstwertigen Bits) im `bits` Feld gelöscht wurden. Das Dokument mit `_id` 2 erfüllt diese Bedingung, da in seinem `bits` Feld diese Bits gelöscht wurden.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$bitsAllClear` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');
```

```
async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('collection');

  const result = await collection.find({
    bits: { $bitsAllClear: 0b0011 }
  }).toArray();

  console.log(result);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['collection']

    result = list(collection.find({
        'bits': { '$bitsAllClear': 0b0011 }
    }))

    print(result)

    client.close()

example()
```

$ Bits AllSet

Der `$bitsAllSet` Operator in Amazon DocumentDB wird verwendet, um Dokumente abzufragen, bei denen ein bestimmter Satz von Bits in einem Feld alle auf 1 gesetzt ist. Dieser Operator ermöglicht es Ihnen, bitweise Operationen an Feldwerten durchzuführen. Er kann nützlich sein, wenn Sie den Status einzelner Bits innerhalb eines numerischen Werts überprüfen müssen.

Parameter

- `field`: Der Name des numerischen Felds, für das die bitweise Operation ausgeführt werden soll.
- `value`: Die numerische Bitmaske, die angibt, welche Bits geprüft werden sollen, oder eine Liste von Bitpositionen, die überprüft werden sollen. Eine numerische Bitmaske kann binär (0b...), dezimal, hexadezimal (0x...), oktal (0o...) oder binär () sein. BinData In einer Liste von Bitpositionen ist die Position des niedrigstwertigen Bits 0.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$bitsAllSet` Operator verwendet wird, um Dokumente zu finden, in denen in dem `flags` Feld alle Bits gesetzt sind, die durch die Bitmaske angegeben sind.

Erstellen Sie Beispieldokumente

```
db.collection.insert([
  { _id: 1, flags: 0b1010 },
  { _id: 2, flags: 0b1100 },
  { _id: 3, flags: 0b1110 }
])
```

Beispiel für eine Abfrage

```
db.collection.find({ flags: { $bitsAllSet: 0b1100 } })
```

Ausgabe

```
{ "_id": 2, "flags": 12 },
{ "_id": 3, "flags": 14 }
```

In diesem Beispiel sucht die Abfrage nach Dokumenten, in denen für das `flags` Feld alle Bits gesetzt sind, die durch die Bitmaske `0b1100` (die den Dezimalwert 12 darstellt) angegeben sind.

Die Dokumente mit `_id` 2 und 3 entsprechen diesen Kriterien, da für ihre `flags` Feldwerte alle erforderlichen Bits gesetzt sind (das dritte und das viertniedrigste Bit).

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$bitsAllSet` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function findWithBitsAllSet() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('collection');

  const result = await collection.find({ flags: { $bitsAllSet:
 0b1100 } }).toArray();
  console.log(result);

  await client.close();
}

findWithBitsAllSet();
```

Python

```
from pymongo import MongoClient

def find_with_bits_all_set():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.collection

    result = list(collection.find({ 'flags': { '$bitsAllSet': 0b1100 } }))
    print(result)

    client.close()
```

```
find_with_bits_all_set()
```

$ Bits AnyClear

Der `$bitsAnyClear` Operator in Amazon DocumentDB wird verwendet, um die Dokumente abzugleichen, bei denen eines der angegebenen Bits in einem Feld gelöscht (auf 0 gesetzt) wurde. Dies kann nützlich sein, um bitweise Operationen an Feldwerten durchzuführen, die in Dokumenten gespeichert sind.

Parameter

- `field`: Das zu prüfende Feld.
- `value`: Die numerische Bitmaske, die angibt, welche Bits geprüft werden sollen, oder eine Liste von Bitpositionen, die überprüft werden sollen. Eine numerische Bitmaske kann binär (0b...), dezimal, hexadezimal (0x...), oktal (0o...) oder binär () sein. BinData In einer Liste von Bitpositionen ist die Position des niedrigstwertigen Bits 0.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$bitsAnyClear` Operator verwendet wird, um zu überprüfen, ob ein Bit im `status` Feld der `items` Sammlung leer ist.

Erstellen Sie Beispieldokumente

```
db.items.insertMany([
  { "_id": 1, "status": 7 },
  { "_id": 2, "status": 15 },
  { "_id": 3, "status": 31 }
]);
```

Beispiel für eine Abfrage

```
db.items.find({ "status": { $bitsAnyClear: 8 } })
```

Ausgabe

```
{ "_id" : 1, "status" : 7 }
```

In diesem Beispiel sucht die Abfrage nach Dokumenten, in denen das `status` Feld Leerzeichen (0) in der Bitmaske 8 (binär`1000`) enthält. Das Dokument mit den `status` Werten 7 (binär`111`) entspricht der Abfrage, da die angegebene Bitmaske mindestens ein Bit Clear enthält. Das passende Clear-Bit ist das viertniedrigstwertige Bit.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$bitsAnyClear` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('items');

  const result = await collection.find({ "status": { $bitsAnyClear:
 8 } }).toArray();
  console.log(result);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.items

    result = list(collection.find({ "status": { "$bitsAnyClear": 8 } }))
```

```
    print(result)

    client.close()

example()
```

$ Bits AnySet

Der `$bitsAnySet` Operator in Amazon DocumentDB wird verwendet, um Dokumente abzufragen, bei denen mindestens ein Bit in den angegebenen Bits in einem Feld auf 1 gesetzt ist. Dieser Operator ermöglicht es Ihnen, bitweise Operationen an den in Feldern gespeicherten Werten durchzuführen, was eine effiziente Abfrage und Analyse von Daten mit bitweisen Eigenschaften ermöglicht.

Parameter

- `field`: Der Feldname, auf den die bitweise Operation angewendet werden soll.
- `value`: Die numerische Bitmaske, die angibt, welche Bits geprüft werden sollen, oder eine Liste von Bitpositionen, die überprüft werden sollen. Eine numerische Bitmaske kann binär (0b...), dezimal, hexadezimal (0x...), oktal (0o...) oder binär () sein. BinData In einer Liste von Bitpositionen ist die Position des niedrigstwertigen Bits 0.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$bitsAnySet` Operator verwendet wird, um Dokumente zu finden, bei denen mindestens ein Bit im `flags` Feld gesetzt ist.

Erstellen Sie Beispieldokumente

```
db.collection.insertMany([
  { _id: 1, flags: 0b1010 },
  { _id: 2, flags: 0b1100 },
  { _id: 3, flags: 0b0011 },
  { _id: 4, flags: 0b0100 }
]);
```

Beispiel für eine Abfrage

```
db.collection.find({
```

```
  flags: { $bitsAnySet: 0b1010 }
})
```

Ausgabe

```
{ "_id" : 1, "flags" : 10 }
{ "_id" : 2, "flags" : 12 }
{ "_id" : 3, "flags" : 3 }
```

Die Abfrage gibt die Dokumente zurück, in denen mindestens eines der in der Bitmaske angegebenen Bits im `flags` Feld gesetzt `0b1010` ist.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$bitsAnySet` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('collection');

  const result = await collection.find({
    flags: { $bitsAnySet: 0b1010 }
  }).toArray();

  console.log(result);

  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient
```

```
client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
db = client['test']
collection = db['collection']

result = list(collection.find({
    'flags': { '$bitsAnySet': 0b1010 }
}))

print(result)

client.close()
```

$comment

Der `$comment` Operator in Amazon DocumentDB wird verwendet, um einen Kommentar an eine Abfrage anzuhängen. Dies kann nützlich sein, um zusätzlichen Kontext oder zusätzliche Informationen zur Abfrage bereitzustellen, was für Debugging- oder Dokumentationszwecke hilfreich sein kann. Der angehängte Kommentar wird als Teil der Ausgabe von Operationen wie db.currentOp () erscheinen.

Parameter

- `string`: Der Kommentar, der der Abfrage angehängt ist.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$comment` Operator in Amazon DocumentDB verwendet wird.

Erstellen Sie Beispieldokumente

```
db.users.insertMany([
  { name: "John Doe", age: 30, email: "john.doe@example.com" },
  { name: "Jane Smith", age: 25, email: "jane.smith@example.com" },
  { name: "Bob Johnson", age: 35, email: "bob.johnson@example.com" }
]);
```

Beispiel für eine Abfrage

```
db.users.find({ age: { $gt: 25 } }, { _id: 0, name: 1, age: 1 }).comment("Retrieve
 users older than 25");
```

Ausgabe

```
{ "name" : "John Doe", "age" : 30 }
{ "name" : "Bob Johnson", "age" : 35 }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$comment` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const users = db.collection('users');

  const result = await users.find({ age: { $gt: 25 } }, { projection: { _id: 0,
 name: 1, age: 1 } })
                            .comment('Retrieve users older than 25')
                            .toArray();

  console.log(result);

  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
```

```
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    users = db.users

    result = list(users.find({ 'age': { '$gt': 25 }}, { '_id': 0, 'name': 1, 'age':
 1 })
                .comment('Retrieve users older than 25'))

    print(result)

    client.close()

if __name__ == '__main__':
    main()
```

$elemMatch

Der `$elemMatch` Operator in Amazon DocumentDB wird verwendet, um ein Array-Feld abzufragen und Dokumente zurückzugeben, bei denen mindestens ein Element im Array den angegebenen Kriterien entspricht. Dieser Operator ist besonders nützlich, wenn Sie komplexe Datenstrukturen mit verschachtelten Arrays oder eingebetteten Dokumenten haben.

Planner Version 2.0 fügte Indexunterstützung für hinzu. `$elemMatch`

Parameter

- `field`: Das abzufragende Array-Feld.
- `query`: Die Kriterien, die mit den Array-Elementen verglichen werden sollen.

Verwendung **`$elemMatch`** innerhalb eines **`$all`** Ausdrucks

Einschränkungen in Bezug auf die Verwendung des `$elemMatch` Operators innerhalb eines `$all` Ausdrucks finden Verwendung von $elemMatch innerhalb eines $all-Ausdrucks Sie unter.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$elemMatch` Operator verwendet wird, um Dokumente zu finden, bei denen das `parts` Array mindestens ein Element enthält, das den angegebenen Kriterien entspricht.

Erstellen Sie Beispieldokumente

```
db.col.insertMany([
  { _id: 1, parts: [{ part: "xyz", qty: 10 }, { part: "abc", qty: 20 }] },
  { _id: 2, parts: [{ part: "xyz", qty: 5 }, { part: "abc", qty: 10 }] },
  { _id: 3, parts: [{ part: "xyz", qty: 15 }, { part: "abc", qty: 100 }] },
  { _id: 4, parts: [{ part: "abc", qty: 150 }] }
]);
```

Beispiel für eine Abfrage

```
db.col.find({
  parts: { "$elemMatch": { part: "xyz", qty: { $lt: 11 } } }
})
```

Ausgabe

```
{ "_id" : 1, "parts" : [ { "part" : "xyz", "qty" : 10 }, { "part" : "abc", "qty" :
 20 } ] }
{ "_id" : 2, "parts" : [ { "part" : "xyz", "qty" : 5 }, { "part" : "abc", "qty" :
 10 } ] }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$elemMatch` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
```

```
  const db = client.db('test');
  const col = db.collection('col');

  const result = await col.find({
    parts: {
      "$elemMatch": { part: "xyz", qty: { $lt: 11 } }
    }
  }).toArray();

  console.log(JSON.stringify(result, null, 2));
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    col = db['col']

    result = list(col.find({
      'parts': {
        '$elemMatch': {'part': 'xyz', 'qty': {'$lt': 11}}
      }
    }))

    print(result)
    client.close()

example()
```

$eq

Der `$eq` Operator in Amazon DocumentDB wird verwendet, um Dokumente abzugleichen, bei denen der Wert eines Felds dem angegebenen Wert entspricht. Dieser Operator wird häufig in der `find()` Methode zum Abrufen von Dokumenten verwendet, die die angegebenen Kriterien erfüllen.

Parameter

- `field`: Das Feld, das auf die Gleichheitsbedingung überprüft werden soll.
- `value`: Der Wert, der mit dem Feld verglichen werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$eq` Operator verwendet wird, um alle Dokumente zu finden, bei denen das `name` Feld gleich ist. `"Thai Curry Palace"`

Erstellen Sie Beispieldokumente

```
db.restaurants.insertMany([
  { name: "Thai Curry Palace", cuisine: "Thai", features: ["Private Dining"] },
  { name: "Italian Bistro", cuisine: "Italian", features: ["Outdoor Seating"] },
  { name: "Mexican Grill", cuisine: "Mexican", features: ["Takeout"] }
]);
```

Beispiel für eine Abfrage

```
db.restaurants.find({ name: { $eq: "Thai Curry Palace" } });
```

Ausgabe

```
{ "_id" : ObjectId("68ee586f916df9d39f3d9414"), "name" : "Thai Curry Palace",
 "cuisine" : "Thai", "features" : [ "Private Dining" ] }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$eq` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function findByName(name) {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
```

```
  const collection = db.collection('restaurants');

  const results = await collection.find({ name: { $eq: name } }).toArray();
  console.log(results);

  await client.close();
}

findByName("Thai Curry Palace");
```

Python

```
from pymongo import MongoClient

def find_by_name(name):
    client = MongoClient("mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false")
    db = client["test"]
    collection = db["restaurants"]

    results = list(collection.find({ "name": { "$eq": name } }))
    print(results)

    client.close()

find_by_name("Thai Curry Palace")
```

$exists

Der `$exists` Operator wird verwendet, um zu überprüfen, ob ein Feld in einem Dokument existiert oder nicht. `$exists`besonders nützlich bei der Arbeit mit Sparse-Indizes in Amazon DocumentDB, bei denen das indizierte Feld möglicherweise nicht in allen Dokumenten vorhanden ist.

Um einen Sparse-Index zu verwenden, den Sie in einer Abfrage erstellt haben, müssen Sie die `$exists`-Bedingungen für die Felder verwenden, die der Index abdecken. Wenn Sie es weglassen`$exists`, verwendet Amazon DocumentDB den Sparse-Index nicht für die Abfrage.

Parameter

- `field`: Der Feldname, dessen Existenz überprüft werden soll.

- `value`: Ein boolescher Wert (`true`oder`false`), der angibt, ob das Feld in den entsprechenden Dokumenten existieren (`true`) oder nicht existieren (`false`) soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung des `$exists` Operators mit einem spärlichen Index für das `special_diets` Feld in der `food` Sammlung.

Erstellen Sie Beispieldokumente

```
db.food.insertMany([
  { _id: 1, name: "Apple", special_diets: ["vegetarian", "gluten-free"] },
  { _id: 2, name: "Broccoli" },
  { _id: 3, name: "Chicken", special_diets: ["dairy-free"] }
]);
```

Erstellen Sie einen spärlichen Index für das Feld `special_diets`

```
db.food.createIndex({ "special_diets": 1 }, { sparse: true, name:
 "special_diets_sparse_asc" });
```

Beispiel für eine Abfrage

```
db.food.find({ "special_diets": { $exists: true } });
```

Ausgabe

```
[
  { "_id" : 1, "name" : "Apple", "special_diets" : [ "vegetarian", "gluten-free" ] },
  { "_id" : 3, "name" : "Chicken", "special_diets" : [ "dairy-free" ] }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$exists` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');
```

```
async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('food');

  const result = await collection.find({ "special_diets": { $exists:
 true } }).toArray();
  console.log(result);

  await client.close();
}

main();
```

Python

```
import pymongo

client = pymongo.MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
db = client['test']
collection = db['food']

result = list(collection.find({"special_diets": {"$exists": True}}))
print(result)

client.close()
```

$expr

Neu ab Version 4.0.

Wird von Elastic Cluster nicht unterstützt.

Der `$expr` Operator in Amazon DocumentDB ermöglicht es Ihnen, Aggregationsausdrücke in der Abfragesprache zu verwenden. Er ermöglicht es Ihnen, komplexe Vergleiche und Berechnungen für Felder innerhalb eines Dokuments durchzuführen, ähnlich wie Sie Aggregationspipeline-Stufen verwenden würden.

Die in einem `$expr` Operator platzierten `$gte` Vergleichsoperatoren `$eq` `$lt` `$lte` `$gt`,, und die Vergleichsoperatoren können einen Index verwenden, einschließlich eines Indexes für die `from` Sammlung, auf die in einer `$lookup` Phase verwiesen wird.

Einschränkungen

- Indizes können nur für Vergleiche zwischen Feldern und Konstanten verwendet werden, daher muss der Operand in eine Konstante aufgelöst werden. Beispielsweise kann ein Vergleich zwischen `$a` und einem konstanten Wert einen Index verwenden, ein Vergleich zwischen `$a` und nicht. `$b`
- Indizes mit mehreren Schlüsseln, Teilindizes oder Indizes mit geringer Dichte werden nicht verwendet. In diesen Fällen fällt die Abfrage auf einen Sammlungsscan zurück.
- Wenn der `let` Operand zu einem leeren oder fehlenden Wert aufgelöst wird, kann für den Vergleich keine gezielte Indexsuche verwendet werden. In diesem Fall führt Amazon DocumentDB einen vollständigen Indexscan durch.

Parameter

- `expression`: Ein Ausdruck, der einen booleschen Wert zurückgibt, sodass Sie Vergleiche und Berechnungen für Dokumentfelder durchführen können.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$expr` Operator verwendet wird, um alle Dokumente zu finden, in denen das `manufacturingCost` Feld größer als das `price` Feld ist.

Erstellen Sie Beispieldokumente

```
db.inventory.insertMany([
  { item: "abc", manufacturingCost: 500, price: 100 },
  { item: "def", manufacturingCost: 300, price: 450 },
  { item: "ghi", manufacturingCost: 400, price: 120 }
]);
```

Beispiel für eine Abfrage

```
db.inventory.find({
  $expr: {
    $gt: ["$manufacturingCost", "$price"]
```

```
  }
})
```

Ausgabe

```
{ "_id" : ObjectId("60b9d4d68d2cac581bc5a89a"), "item" : "abc", "manufacturingCost" :
 500, "price" : 100 },
{ "_id" : ObjectId("60b9d4d68d2cac581bc5a89c"), "item" : "ghi", "manufacturingCost" :
 400, "price" : 120 }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$expr` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('inventory');

  const result = await collection.find({
    $expr: {
      $gt: ['$manufacturingCost', '$price']
    }
  }).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient
```

```
def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['inventory']

    result = list(collection.find({
        '$expr': {
            '$gt': ['$manufacturingCost', '$price']
        }
    }))

    print(result)
    client.close()

example()
```

$gt

Der `$gt` Operator in Amazon DocumentDB wird verwendet, um Dokumente auszuwählen, bei denen der Wert des angegebenen Felds größer als der angegebene Wert ist. Dieser Operator ist nützlich, um Daten auf der Grundlage numerischer Vergleiche zu filtern und abzufragen.

Parameter

- `field`: Das zu vergleichende Feld.
- `value`: Der Wert, mit dem verglichen werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$gt` Operator verwendet wird, um alle Dokumente zu finden, bei denen das `age` Feld größer als 30 ist.

Erstellen Sie Beispieldokumente

```
db.users.insertMany([
  { name: "John", age: 25 },
  { name: "Jane", age: 32 },
  { name: "Bob", age: 45 },
  { name: "Alice", age: 28 }
```

```
]);
```

Beispiel für eine Abfrage

```
db.users.find({ age: { $gt: 30 } });
```

Ausgabe

```
{ "_id" : ObjectId("6249e5c22a5d39884a0a0001"), "name" : "Jane", "age" : 32 },
{ "_id" : ObjectId("6249e5c22a5d39884a0a0002"), "name" : "Bob", "age" : 45 }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$gt` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function findUsersOlderThan30() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const users = await db.collection('users').find({ age: { $gt: 30 } }).toArray();
  console.log(users);
  await client.close();
}

findUsersOlderThan30();
```

Python

```
from pymongo import MongoClient

def find_users_older_than_30():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    users = list(db.users.find({ 'age': { '$gt': 30 } }))
```

```
    print(users)
    client.close()

find_users_older_than_30()
```

$gte

Der `$gte` Operator in Amazon DocumentDB wird verwendet, um Werte abzugleichen, die größer oder gleich einem bestimmten Wert sind. Dieser Operator ist nützlich, um Daten auf der Grundlage numerischer Vergleiche zu filtern und abzufragen.

Parameter

- `field`: Das Feld, das mit dem angegebenen Wert verglichen werden soll.
- `value`: Der Wert, der mit dem Feld verglichen werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung des `$gte` Operators in Amazon DocumentDB, um alle Dokumente zu finden, bei denen das Feld „Alter“ größer oder gleich 25 ist.

Erstellen Sie Beispieldokumente

```
db.users.insertMany([
  { _id: 1, name: "John", age: 20 },
  { _id: 2, name: "Jane", age: 25 },
  { _id: 3, name: "Bob", age: 30 },
  { _id: 4, name: "Alice", age: 35 }
]);
```

Beispiel für eine Abfrage

```
db.users.find({ age: { $gte: 25 } }, { _id: 0, name: 1, age: 1 });
```

Ausgabe

```
{ "name" : "Jane", "age" : 25 }
{ "name" : "Bob", "age" : 30 }
{ "name" : "Alice", "age" : 35 }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$gte` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function findUsersAboveAge(age) {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const users = db.collection('users');

  const result = await users.find({ age: { $gte: age } }, { projection: { _id: 0,
 name: 1, age: 1 } }).toArray();
  console.log(result);

  await client.close();
}

findUsersAboveAge(25);
```

Python

```
from pymongo import MongoClient

def find_users_above_age(age):
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    users = db.users

    result = list(users.find({ 'age': { '$gte': age } }, { '_id': 0, 'name': 1,
 'age': 1 }))
    print(result)

    client.close()

find_users_above_age(25)
```

$in

Der `$in` Operator in Amazon DocumentDB ist ein logischer Abfrageoperator, mit dem Sie Dokumente finden können, bei denen der Wert eines Felds einem der in einem Array angegebenen Werte entspricht.

Parameter

- `field`: Das Feld, das mit dem bereitgestellten Array verglichen werden soll.
- `[value1, value2, ...]`: Ein Array von Werten, die mit dem angegebenen Feld verglichen werden sollen.

Dollar (**$**) in Feldnamen

Informationen Dollar ($) und Punkt (.) in Feldnamen zu Einschränkungen beim Abfragen von Feldern mit $ Präfix `$in` in verschachtelten Objekten finden Sie unter.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$in` Operator verwendet wird, um Dokumente zu finden, bei denen das `color` Feld einem der Werte im bereitgestellten Array entspricht.

Erstellen Sie Beispieldokumente

```
db.colors.insertMany([
  { "_id": 1, "color": "red" },
  { "_id": 2, "color": "green" },
  { "_id": 3, "color": "blue" },
  { "_id": 4, "color": "yellow" },
  { "_id": 5, "color": "purple" }
])
```

Beispiel für eine Abfrage

```
db.colors.find({ "color": { "$in": ["red", "blue", "purple"] } })
```

Ausgabe

```
{ "_id": 1, "color": "red" },
```

```
{ "_id": 3, "color": "blue" },
{ "_id": 5, "color": "purple" }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$in` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function findByIn() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('colors');

  const result = await collection.find({ "color": { "$in": ["red", "blue",
 "purple"] } }).toArray();
  console.log(result);

  await client.close();
}

findByIn();
```

Python

```
from pymongo import MongoClient

def find_by_in():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.colors

    result = list(collection.find({ "color": { "$in": ["red", "blue",
 "purple"] } }))
    print(result)
```

```
    client.close()

find_by_in()
```

$jsonSchema

Neu ab Version 4.0.

Wird von Elastic Cluster nicht unterstützt.

Der `$jsonSchema` Operator in Amazon DocumentDB wird verwendet, um Dokumente auf der Grundlage eines bestimmten JSON-Schemas zu filtern. Mit diesem Operator können Sie Dokumente abfragen, die einem bestimmten JSON-Schema entsprechen, und so sicherstellen, dass die abgerufenen Dokumente bestimmten Struktur- und Datentypanforderungen entsprechen.

Wenn Sie den `$jsonSchema` Bewertungsabfrageoperator als Teil einer Sammlungserstellung verwenden, können Sie das Schema der Dokumente überprüfen, die in die Sammlung eingefügt werden. Verwenden der JSON-SchemavalidierungWeitere Informationen finden Sie unter.

Parameter

- `required`(Array): Gibt die erforderlichen Felder im Dokument an.
- `properties`(Objekt): Definiert den Datentyp und andere Einschränkungen für jedes Feld im Dokument.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$jsonSchema` Operator verwendet wird, um die `employees` Sammlung so zu filtern, dass nur Dokumente abgerufen werden`name`, die die `age` Felder `employeeId` und enthalten und das `employeeId` Feld vom Typ `string` ist.

Erstellen Sie Beispieldokumente

```
db.employees.insertMany([
  { "name": { "firstName": "Carol", "lastName": "Smith" }, "employeeId": "1" },
  { "name": { "firstName": "Emily", "lastName": "Brown" }, "employeeId": "2", "age":
 25 },
  { "name": { "firstName": "William", "lastName": "Taylor" }, "employeeId": 3, "age":
 24 },
```

```
  { "name": { "firstName": "Jane", "lastName": "Doe" }, "employeeId": "4" }
]);
```

Beispiel für eine Abfrage

```
db.employees.aggregate([
  { $match: {
    $jsonSchema: {
      required: ["name", "employeeId", "age"],
      properties: { "employeeId": { "bsonType": "string" } }
    }
  }}
]);
```

Ausgabe

```
{ "_id" : ObjectId("6908e8b61f77fc26b2ecd26f"), "name" : { "firstName" : "Emily",
 "lastName" : "Brown" }, "employeeId" : "2", "age" : 25 }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$jsonSchema` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function filterByJsonSchema() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('employees');

  const result = await collection.aggregate([
    {
      $match: {
        $jsonSchema: {
          required: ['name', 'employeeId', 'age'],
          properties: { 'employeeId': { 'bsonType': 'string' } }
        }
```

```
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

filterByJsonSchema();
```

Python

```
from pymongo import MongoClient

def filter_by_json_schema():
  client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
  db = client['test']
  collection = db['employees']

  result = list(collection.aggregate([
    {
      '$match': {
        '$jsonSchema': {
          'required': ['name', 'employeeId', 'age'],
          'properties': {'employeeId': {'bsonType': 'string'}}
        }
      }
    }
  ]))

  print(result)
  client.close()

filter_by_json_schema()
```

$lt

Der `$lt` Operator in Amazon DocumentDB wird verwendet, um Werte abzugleichen, die unter dem angegebenen Wert liegen. Dieser Operator ist nützlich, um Daten auf der Grundlage numerischer Vergleiche zu filtern und abzufragen.

Parameter

- `field`: Der zu prüfende Feldname.
- `value`: Der Wert, mit dem verglichen werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$lt` Operator verwendet wird, um Dokumente abzurufen, deren `Inventory.OnHand` Wert kleiner als 50 ist.

Erstellen Sie Beispieldokumente

```
db.example.insertMany([
  { "_id": 1, "Inventory": { "OnHand": 25, "Sold": 60 }, "Colors": ["Red", "Blue"] },
  { "_id": 2, "Inventory": { "OnHand": 75, "Sold": 40 }, "Colors": ["Red", "White"] },
  { "_id": 3, "Inventory": { "OnHand": 10, "Sold": 85 }, "Colors": ["Red", "Green"] }
]);
```

Beispiel für eine Abfrage

```
db.example.find({ "Inventory.OnHand": { $lt: 50 } })
```

Ausgabe

```
{ "_id" : 1, "Inventory" : { "OnHand" : 25, "Sold" : 60 }, "Colors" : [ "Red",
 "Blue" ] }
{ "_id" : 3, "Inventory" : { "OnHand" : 10, "Sold" : 85 }, "Colors" : [ "Red",
 "Green" ] }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$lt` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
```

```
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('example');

  const result = await collection.find({ "Inventory.OnHand": { $lt:
 50 } }).toArray();
  console.log(result);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['example']

    result = list(collection.find({ "Inventory.OnHand": { "$lt": 50 } }))
    print(result)

    client.close()

example()
```

$mod

Der $mod Abfrageoperator wählt Dokumente aus, in denen ein durch einen Divisor geteilter Feldwert einen angegebenen Rest hat. Dies ist nützlich, um Dokumente auf der Grundlage modularer arithmetischer Bedingungen zu filtern.

Parameter

- `divisor`: Die Zahl, durch die dividiert werden soll.

- `remainder`: Der erwartete Restwert.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der $mod Operator verwendet wird, um alle Bestellungen zu finden, bei denen die Menge eine ungerade Zahl ist.

Erstellen Sie Beispieldokumente

```
db.orders.insertMany([
  { _id: 1, item: "Widget", quantity: 15 },
  { _id: 2, item: "Gadget", quantity: 20 },
  { _id: 3, item: "Tool", quantity: 7 },
  { _id: 4, item: "Device", quantity: 12 },
  { _id: 5, item: "Part", quantity: 9 }
]);
```

Beispiel für eine Abfrage

```
db.orders.find({ quantity: { $mod: [2, 1] } });
```

Ausgabe

```
{ "_id" : 1, "item" : "Widget", "quantity" : 15 }
{ "_id" : 3, "item" : "Tool", "quantity" : 7 }
{ "_id" : 5, "item" : "Part", "quantity" : 9 }
```

Diese Abfrage gibt Dokumente zurück, bei denen die Menge geteilt durch 2 einen Rest von 1 hat, wodurch praktisch alle ungeraden Mengen ausgewählt werden.

Codebeispiele

Um ein Codebeispiel für die Verwendung des $mod Abfrageoperators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');
```

```
async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('orders');

  const result = await collection.find({ quantity: { $mod: [2, 1] } }).toArray();

  console.log(JSON.stringify(result, null, 2));
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['orders']

    result = list(collection.find({'quantity': {'$mod': [2, 1]}}))

    print(result)
    client.close()

example()
```

$lte

Der `$lte` Operator in Amazon DocumentDB wird verwendet, um Dokumente abzugleichen, bei denen der Wert eines angegebenen Felds kleiner oder gleich dem angegebenen Wert ist. Dieser Operator ist nützlich, um Daten auf der Grundlage numerischer Vergleiche zu filtern und abzufragen.

Parameter

- `field`: Das zu vergleichende Feld.

- `value`: Der Wert, mit dem verglichen werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung des `$lte` Operators zum Abrufen von Dokumenten, bei denen das `quantity` Feld kleiner oder gleich 10 ist.

Erstellen Sie Beispieldokumente

```
db.inventory.insertMany([
  { item: "canvas", qty: 100 },
  { item: "paint", qty: 50 },
  { item: "brush", qty: 10 },
  { item: "paper", qty: 5 }
]);
```

Beispiel für eine Abfrage

```
db.inventory.find({ qty: { $lte: 10 } });
```

Ausgabe

```
{ "_id" : ObjectId("..."), "item" : "brush", "qty" : 10 },
{ "_id" : ObjectId("..."), "item" : "paper", "qty" : 5 }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$lte` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require("mongodb");

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db("test");
```

```
  const collection = db.collection("inventory");

  const result = await collection.find({ qty: { $lte: 10 } }).toArray();
  console.log(result);

  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client["test"]
    collection = db["inventory"]

    result = list(collection.find({ "qty": { "$lte": 10 } }))
    print(result)

    client.close()

if __name__ == "__main__":
    main()
```

$meta

Der `$meta` Operator wird verwendet, um auf Metadaten zuzugreifen, die mit der aktuellen Abfrageausführung verknüpft sind. Dieser Operator wird hauptsächlich für Textsuchoperationen verwendet, bei denen die Metadaten Informationen über die Relevanz der übereinstimmenden Dokumente liefern können.

Parameter

- `textScore`: Ruft das Ergebnis der Textsuche für das Dokument ab. Diese Bewertung gibt die Relevanz des Dokuments für die Textsuchabfrage an.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$meta` Operator verwendet wird, um das Ergebnis der Textsuche für Dokumente abzurufen, die einer Textsuchabfrage entsprechen.

Erstellen Sie Beispieldokumente

```
db.documents.insertMany([
  { _id: 1, title: "Coffee Basics", content: "Coffee is a popular beverage made from
 roasted coffee beans." },
  { _id: 2, title: "Coffee Culture", content: "Coffee coffee coffee - the ultimate
 guide to coffee brewing and coffee preparation." },
  { _id: 3, title: "Tea vs Coffee", content: "Many people prefer tea over coffee for
 its health benefits." }
]);
```

Textindex erstellen

```
db.documents.createIndex({ content: "text" });
```

Beispiel für eine Abfrage

```
db.documents.find(
  { $text: { $search: "coffee" } },
  { _id: 0, title: 1, content: 1, score: { $meta: "textScore" } }
).sort({ score: { $meta: "textScore" } });
```

Ausgabe

```
[
  {
    title: 'Coffee Culture',
    content: 'Coffee coffee coffee - the ultimate guide to coffee brewing and coffee
 preparation.',
    score: 0.8897688388824463
  },
  {
    title: 'Coffee Basics',
    content: 'Coffee is a popular beverage made from roasted coffee beans.',
    score: 0.75990891456604
  },
```

```
  {
    title: 'Tea vs Coffee',
    content: 'Many people prefer tea over coffee for its health benefits.',
    score: 0.6079270839691162
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$meta` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function findWithTextScore() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('documents');

  const result = await collection.find(
    { $text: { $search: "coffee" } },
    { projection: { _id: 0, title: 1, content: 1, score: { $meta: "textScore" } } }
  ).sort({ score: { $meta: "textScore" } }).toArray();

  console.log(result);
  client.close();
}

findWithTextScore();
```

Python

```
from pymongo import MongoClient

client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
db = client['test']
collection = db['documents']
```

```
for doc in collection.find(
    {'$text': {'$search': 'coffee'}},
    {'_id': 0, 'title': 1, 'content': 1, 'score': {'$meta': 'textScore'}}
).sort([('score', {'$meta': 'textScore'})]):
    print(doc)

client.close()
```

$ne

Der `$ne` Operator wird verwendet, um Dokumente abzugleichen, bei denen der Wert eines Felds nicht dem angegebenen Wert entspricht. Es ist ein Vergleichsoperator, der in Abfrageprädikaten zum Filtern von Dokumenten verwendet werden kann.

Planner Version 2.0 fügte Indexunterstützung für hinzu. `$ne`

Parameter

- `field`: Das zu prüfende Feld.
- `value`: Der Wert, gegen den geprüft werden soll.

Beispiel (MongoDB Shell)

In diesem Beispiel finden wir alle Dokumente in der `users` Sammlung, in denen das `status` Feld ungleich ist. `"active"`

Erstellen Sie Beispieldokumente

```
db.users.insertMany([
  { name: "John", status: "active" },
  { name: "Jane", status: "inactive" },
  { name: "Bob", status: "suspended" },
  { name: "Alice", status: "active" }
]);
```

Beispiel für eine Abfrage

```
db.users.find({ status: { $ne: "active" } });
```

Ausgabe

```
[
  {
    _id: ObjectId('...'),
    name: 'Jane',
    status: 'inactive'
  },
  {
    _id: ObjectId('...'),
    name: 'Bob',
    status: 'suspended'
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des $ne Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const users = db.collection('users');

  const result = await users.find({ status: { $ne: 'active' } }).toArray();
  console.log(result);

  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient
```

```
client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
db = client['test']
users = db['users']

result = list(users.find({ 'status': { '$ne': 'active' } }))
print(result)

client.close()
```

$nin

Der `$nin` Operator wird verwendet, um Werte abzugleichen, die sich nicht im angegebenen Array befinden. Es ist die Umkehrung des `$in` Operators, der mit Werten übereinstimmt, die sich im angegebenen Array befinden.

Planner Version 2.0 fügte Indexunterstützung für `$nin` hinzu.

Parameter

- `field`: Das zu überprüfende Feld.
- `array`: Das Array von Werten, anhand dessen geprüft werden soll.

Dollar (**$**) in Feldnamen

Einschränkungen Dollar ($) und Punkt (.) in Feldnamen bei der Abfrage von Feldern mit $ Präfix `$nin` in verschachtelten Objekten finden Sie unter.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$nin` Operator verwendet wird, um Dokumente zu finden, bei denen das `category` Feld nicht „Fiction“ oder „Mystery“ entspricht.

Erstellen Sie Beispieldokumente

```
db.books.insertMany([
```

```
  { title: "The Great Gatsby", author: "F. Scott Fitzgerald", category: "Fiction" },
  { title: "To Kill a Mockingbird", author: "Harper Lee", category: "Fiction" },
  { title: "The Girl on the Train", author: "Paula Hawkins", category: "Mystery" },
  { title: "The Martian", author: "Andy Weir", category: "Science Fiction" },
  { title: "The Alchemist", author: "Paulo Coelho", category: "Philosophy" }
])
```

Beispiel für eine Abfrage

```
db.books.find({
  category: {
    $nin: ["Fiction", "Mystery"]
  }
})
```

Ausgabe

```
[
  {
    _id: ObjectId('...'),
    title: 'The Martian',
    author: 'Andy Weir',
    category: 'Science Fiction'
  },
  {
    _id: ObjectId('...'),
    title: 'The Alchemist',
    author: 'Paulo Coelho',
    category: 'Philosophy'
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$nin` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');
```

```
async function findBooksNotInCategories(categories) {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const books = await db.collection('books').find({
    category: {
      $nin: categories
    }
  }).toArray();
  console.log(books);
  client.close();
}

findBooksNotInCategories(['Fiction', 'Mystery']);
```

Python

```
from pymongo import MongoClient

def find_books_not_in_categories(categories):
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    books = list(db.books.find({
        'category': {
            '$nin': categories
        }
    }))
    print(books)
    client.close()

find_books_not_in_categories(['Fiction', 'Mystery'])
```

$nor

Der `$nor` Operator wird verwendet, um Dokumente abzugleichen, bei denen keine der angegebenen Abfragebedingungen zutrifft. Er ähnelt der logischen Operation „NOR“, bei der das Ergebnis wahr ist, wenn keiner der Operanden wahr ist.

Parameter

- `expression1`: Der erste Ausdruck, der ausgewertet werden soll.
- `expression2`: Der zweite auszuwertende Ausdruck.
- `expressionN`: Zusätzliche auszuwertende Ausdrücke.

Beispiel (MongoDB Shell)

Das folgende Beispiel demonstriert die Verwendung des `$nor` Operators, indem Dokumente abgerufen werden, bei denen das `qty` Feld nicht weniger als 20 ist und das `size` Feld nicht „XL“ entspricht.

Erstellen Sie Beispieldokumente

```
db.items.insertMany([
  { qty: 10, size: "M" },
  { qty: 15, size: "XL" },
  { qty: 25, size: "L" },
  { qty: 30, size: "XL" }
])
```

Beispiel für eine Abfrage

```
db.items.find({
  $nor: [
    { qty: { $lt: 20 } },
    { size: "XL" }
  ]
})
```

Ausgabe

```
[
  { "_id" : ObjectId("..."), "qty" : 25, "size" : "L" }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$nor` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('items');

  const result = await collection.find({
    $nor: [
      { qty: { $lt: 20 } },
      { size: "XL" }
    ]
  }).toArray();

  console.log(result);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['items']

    result = list(collection.find({
        '$nor': [
            { 'qty': { '$lt': 20 } },
            { 'size': 'XL' }
        ]
    }))
```

```
    print(result)

    client.close()

example()
```

$not

Der `$not` Operator wird verwendet, um das Ergebnis eines bestimmten Ausdrucks zu negieren. Damit können Sie Dokumente auswählen, bei denen die angegebene Bedingung nicht erfüllt ist.

Planner Version 2.0 hat Indexunterstützung für `$not {eq}` und hinzugefügt`$not {in}`.

Parameter

- `expression`: Der Ausdruck, der negiert werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$not` Operator verwendet wird, um Dokumente zu finden, bei denen das `status` Feld nicht „aktiv“ entspricht.

Erstellen Sie Beispieldokumente

```
db.users.insertMany([
  { name: "John", status: "active" },
  { name: "Jane", status: "inactive" },
  { name: "Bob", status: "pending" },
  { name: "Alice", status: "active" }
]);
```

Beispiel für eine Abfrage

```
db.users.find({ status: { $not: { $eq: "active" } } });
```

Ausgabe

```
[
  {
```

```
    _id: ObjectId('...'),
    name: 'Jane',
    status: 'inactive'
  },
  {
    _id: ObjectId('...'),
    name: 'Bob',
    status: 'pending'
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$not` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient, Filters } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('users');

  const result = await collection.find({
    status: { $not: { $eq: "active" } }
  }).toArray();

  console.log(result);

  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient
from bson.son import SON
```

```
client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
db = client['test']
collection = db['users']

result = collection.find({
    "status": {"$not": {"$eq": "active"}}
})

for doc in result:
    print(doc)

client.close()
```

$or

Der `$or` Operator wird verwendet, um eine logische ODER-Operation für ein Array aus zwei oder mehr Ausdrücken auszuführen. Er gibt Dokumente zurück, die mindestens einem der Ausdrücke entsprechen. Dieser Operator ist nützlich, wenn Sie nach Dokumenten abfragen müssen, die eine von mehreren Bedingungen erfüllen.

Parameter

- `expression1`: Der erste auszuwertende Ausdruck.
- `expression2`: Der zweite auszuwertende Ausdruck.
- `...`: Zusätzliche Ausdrücke, die ausgewertet werden sollen (optional).

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$or` Operator verwendet `make` wird, um Dokumente zu finden, in denen der entweder "TruckForYou" mit dem Modell „Heavy H1" oder" SportForYou "mit dem Modell „Bolid 1" ist.

Erstellen Sie Beispieldokumente

```
db.cars.insertMany([
  { make: "TruckForYou", model: "Heavy H1", year: 2020 },
  { make: "SportForYou", model: "Bolid 1", year: 2021 },
  { make: "TruckForYou", model: "Cargo 5", year: 2019 },
```

```
  { make: "SportForYou", model: "Racer 2", year: 2022 }
]);
```

Beispiel für eine Abfrage

```
db.cars.find({
  $or: [
    { make: "TruckForYou", model: "Heavy H1" },
    { make: "SportForYou", model: "Bolid 1" }
  ]
});
```

Ausgabe

```
[
  {
    _id: ObjectId('...'),
    make: 'TruckForYou',
    model: 'Heavy H1',
    year: 2020
  },
  {
    _id: ObjectId('...'),
    make: 'SportForYou',
    model: 'Bolid 1',
    year: 2021
  }
]
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$or` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function findCarsByMakeModel() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
```

```
  const db = client.db('test');
  const cars = db.collection('cars');

  const result = await cars.find({
    $or: [
      { make: "TruckForYou", model: "Heavy H1" },
      { make: "SportForYou", model: "Bolid 1" }
    ]
  }).toArray();

  console.log(result);
  client.close();
}

findCarsByMakeModel();
```

Python

```
from pymongo import MongoClient

def find_cars_by_make_model():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    cars = db.cars

    result = list(cars.find({
        '$or': [
            {'make': 'TruckForYou', 'model': 'Heavy H1'},
            {'make': 'SportForYou', 'model': 'Bolid 1'}
        ]
    }))

    print(result)
    client.close()

find_cars_by_make_model()
```

$regex

Mit dem `$regex` Operator können Sie einen Abgleich mit regulären Ausdrücken für Zeichenkettenfelder durchführen. Es ist ein leistungsstarkes Tool zum Suchen und Filtern von Dokumenten, die auf komplexen Mustern basieren.

Parameter

- `regular expression`: Das reguläre Ausdrucksmuster, das mit dem Feld abgeglichen werden soll.
- `$options`: (optional) Stellt Optionen zur Änderung des Suchverhaltens bereit, z. B. Berücksichtigung von Groß- und Kleinschreibung, globaler Abgleich usw.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$regex` Operator verwendet wird, um nach Dokumenten zu suchen, bei denen das Feld „Name“ einem bestimmten Muster entspricht.

Erstellen Sie Beispieldokumente

```
db.users.insertMany([
  { name: "John Doe" },
  { name: "Jane Smith" },
  { name: "Alice Johnson" },
  { name: "Bob Williams" },
  { name: "Charlie Davis" }
]);
```

Beispiel für eine Abfrage

```
db.users.find({ name: { $regex: /^A/ } })
```

Ausgabe

```
[
  { "_id" : ObjectId("..."), "name" : "Alice Johnson" }
]
```

Diese Abfrage gibt alle Dokumente zurück, bei denen das Feld „Name“ mit dem Buchstaben „A“ beginnt.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$regex` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('users');

  const results = await collection.find({ name: { $regex: /^A/ } }).toArray();
  console.log(results);

  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['users']

    results = list(collection.find({ 'name': { '$regex': '^A' } }))
    print(results)

    client.close()

if __name__ == "__main__":
    main()
```

$ Beispielrate

Neu ab Version 8.0.1.

Wird von Elastic Cluster nicht unterstützt.

Der `$sampleRate` Operator in Amazon DocumentDB vergleicht eine Zufallsstichprobe der Eingabedokumente auf der Grundlage der angegebenen Rate. Sie können ihn in einem `find()` Abfragefilter oder in einer `$match` Aggregationsphase verwenden. Da die Auswahl probabilistisch ist, handelt es sich bei der Anzahl der zurückgegebenen Dokumente um einen Näherungswert, der je nach Durchlauf variieren kann.

Parameter

- `value`: Eine Zahl im Bereich `[0, 1]` (einschließlich), die direkt an den Operator übergeben wird (z. B.`{ $sampleRate: 0.3 }`), die die Wahrscheinlichkeit festlegt, mit der jedes Dokument eingeschlossen wird. Ein Wert von `0` gibt keine Dokumente zurück; ein Wert von 1 gibt alle Dokumente zurück.

Beispiel (MongoDB Shell)

Das folgende Beispiel verwendet`$sampleRate`, um ungefähr 30 Prozent der Dokumente in einer Sammlung zurückzugeben.

Erstellen Sie Beispieldokumente

```
db.events.insertMany([
  { _id: 1, type: "click" },
  { _id: 2, type: "view" },
  { _id: 3, type: "click" },
  { _id: 4, type: "view" },
  { _id: 5, type: "purchase" }
]);
```

Beispiel für eine Abfrage

```
db.events.aggregate([
  { $match: { $sampleRate: 0.3 } }
]);
```

Ausgabe

Die Operation gibt eine zufällige Teilmenge der Dokumente zurück. Da `$sampleRate` es probabilistisch ist, variieren die spezifischen zurückgegebenen Dokumente und ihre Anzahl bei jeder Ausführung der Abfrage.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$sampleRate` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const uri = 'mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false';
  const client = new MongoClient(uri);

  try {
    await client.connect();
    const db = client.db('test');
    const collection = db.collection('events');

    // Return approximately 30% of the documents
    const result = await collection.aggregate([
      { $match: { $sampleRate: 0.3 } }
    ]).toArray();

    console.log(result);
  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient
```

```
def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    try:
        db = client['test']
        collection = db['events']

        # Return approximately 30% of the documents
        result = list(collection.aggregate([
            { '$match': { '$sampleRate': 0.3 } }
        ]))

        print(result)
    finally:
        client.close()

example()
```

$slice

Der `$slice` Projektionsoperator begrenzt die Anzahl der Array-Elemente, die in einem Abfrageergebnis zurückgegeben werden. Er ermöglicht es Ihnen, eine bestimmte Anzahl von Elementen vom Anfang oder Ende eines Array-Felds abzurufen, ohne das gesamte Array zu laden.

Parameter

- `field`: Das zu projizierende Array-Feld.
- `count`: Anzahl der zurückzugebenden Elemente. Positive Werte geben Elemente vom Anfang zurück, negative Werte vom Ende.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$slice` Projektionsoperator verwendet wird, um nur die ersten beiden Elemente aus einem Array-Feld zurückzugeben.

Erstellen Sie Beispieldokumente

```
db.inventory.insertMany([
  { _id: 1, item: "notebook", tags: ["office", "school", "supplies", "writing"] },
```

```
  { _id: 2, item: "pen", tags: ["office", "writing"] },
  { _id: 3, item: "folder", tags: ["office", "supplies", "storage", "organization"] }
]);
```

Beispiel für eine Abfrage

```
db.inventory.find(
  {},
  { item: 1, tags: { $slice: 2 } }
)
```

Ausgabe

```
{ "_id" : 1, "item" : "notebook", "tags" : [ "office", "school" ] }
{ "_id" : 2, "item" : "pen", "tags" : [ "office", "writing" ] }
{ "_id" : 3, "item" : "folder", "tags" : [ "office", "supplies" ] }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$slice` Projektionsoperators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('inventory');

  const result = await collection.find(
    {},
    { projection: { item: 1, tags: { $slice: 2 } } }
  ).toArray();

  console.log(JSON.stringify(result, null, 2));
  await client.close();
}
```

```
example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['inventory']

    result = list(collection.find(
        {},
        {'item': 1, 'tags': {'$slice': 2}}
    ))

    print(result)
    client.close()

example()
```

$size

Der `$size` Abfrageoperator sucht nach Dokumenten, in denen ein Array-Feld genau die angegebene Anzahl von Elementen hat. Dies ist nützlich, um Dokumente anhand der Array-Länge zu filtern.

Parameter

- `field`: Das zu prüfende Array-Feld.
- `count`: Die genaue Anzahl der Elemente, die das Array enthalten muss.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$size` Operator verwendet wird, um alle Produkte zu finden, die genau drei Tags haben.

Erstellen Sie Beispieldokumente

```
db.products.insertMany([
```

```
  { _id: 1, name: "Laptop", tags: ["electronics", "computers", "portable"] },
  { _id: 2, name: "Mouse", tags: ["electronics", "accessories"] },
  { _id: 3, name: "Desk", tags: ["furniture", "office", "workspace"] },
  { _id: 4, name: "Monitor", tags: ["electronics"] }
]);
```

Beispiel für eine Abfrage

```
db.products.find({ tags: { $size: 3 } });
```

Ausgabe

```
{ "_id" : 1, "name" : "Laptop", "tags" : [ "electronics", "computers", "portable" ] }
{ "_id" : 3, "name" : "Desk", "tags" : [ "furniture", "office", "workspace" ] }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$size` Abfrageoperators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('products');

  const result = await collection.find({ tags: { $size: 3 } }).toArray();

  console.log(JSON.stringify(result, null, 2));
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient
```

```
def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['products']

    result = list(collection.find({'tags': {'$size': 3}}))

    print(result)
    client.close()

example()
```

$text

Der `$text` Operator wird verwendet, um eine Volltextsuche in textindizierten Feldern innerhalb einer Dokumentsammlung durchzuführen. Mit diesem Operator können Sie nach Dokumenten suchen, die bestimmte Wörter oder Ausdrücke enthalten. Er kann mit anderen Abfrageoperatoren kombiniert werden, um Ergebnisse anhand zusätzlicher Kriterien zu filtern.

Parameter

- `$search`: Die Textzeichenfolge, nach der gesucht werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie Sie den `$text` Operator verwenden, um nach Dokumenten zu suchen, die das Wort „Interesse“ enthalten, und die Ergebnisse anhand eines Felds „star_rating“ zu filtern.

Erstellen Sie Beispieldokumente

```
db.test.insertMany([
  { "_id": 1, "star_rating": 4, "comments": "apple is red" },
  { "_id": 2, "star_rating": 5, "comments": "pie is delicious" },
  { "_id": 3, "star_rating": 3, "comments": "apples, oranges - healthy fruit" },
  { "_id": 4, "star_rating": 2, "comments": "bake the apple pie in the oven" },
  { "_id": 5, "star_rating": 5, "comments": "interesting couch" },
  { "_id": 6, "star_rating": 5, "comments": "interested in couch for sale, year 2022" }
```

```
]);
```

Textindex erstellen

```
db.test.createIndex({ comments: "text" });
```

Beispiel für eine Abfrage

```
db.test.find({$and: [{star_rating: 5}, {$text: {$search: "interest"}}]})
```

Ausgabe

```
{ "_id" : 5, "star_rating" : 5, "comments" : "interesting couch" }
{ "_id" : 6, "star_rating" : 5, "comments" : "interested in couch for sale, year
 2022" }
```

Der obige Befehl gibt Dokumente mit einem textindizierten Feld zurück, das jede Form von „Zinsen“ enthält und deren „Star_Rating“ dem Wert 5 entspricht.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$text` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function searchDocuments() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('test');

  const result = await collection.find({
    $and: [
      { star_rating: 5 },
      { $text: { $search: 'interest' } }
    ]
  }).toArray();
```

```
  console.log(result);
  client.close();
}

searchDocuments();
```

Python

```
from pymongo import MongoClient

def search_documents():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db.test

    result = list(collection.find({
        '$and': [
            {'star_rating': 5},
            {'$text': {'$search': 'interest'}}
        ]
    }))

    print(result)
    client.close()

search_documents()
```

$type

Der `$type` Operator wird verwendet, um den Datentyp eines Felds in einem Dokument zu überprüfen. Er kann verwendet werden, wenn typspezifische Operationen oder Validierungen erforderlich sind. Der `$type` Operator gibt den BSON-Typ des ausgewerteten Ausdrucks zurück. Der zurückgegebene Typ ist eine Zeichenfolge, die dem Typ des Felds oder Ausdrucks entspricht.

Planner Version 2.0 fügte Indexunterstützung für hinzu`$type`.

Parameter

- `expression`: Der auszuwertende Ausdruck.

Beispiel (MongoDB Shell)

Erstellen Sie Beispieldokumente

```
db.documents.insertMany([
  { _id: 1, name: "John", age: 30, email: "john@example.com" },
  { _id: 2, name: "Jane", age: "25", email: 123456 },
  { _id: 3, name: 123, age: true, email: null }
]);
```

Beispiel für eine Abfrage

```
db.documents.find({
  $or: [
    { age: { $type: "number" } },
    { email: { $type: "string" } },
    { name: { $type: "string" } }
  ]
})
```

Ausgabe

```
[
  { "_id": 1, "name": "John", "age": 30, "email": "john@example.com" },
  { "_id": 2, "name": "Jane", "age": "25", "email": 123456 }
]
```

Diese Abfrage gibt die Dokumente zurück, in denen das `age` Feld vom Typ „Zahl“, das `email` Feld vom Typ „Zeichenfolge“ und das `name` Feld vom Typ „Zeichenfolge“ ist.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$type` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function findByType() {
```

```
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('documents');

  const results = await collection.find({
    $or: [
      { age: { $type: 'number' } },
      { email: { $type: 'string' } },
      { name: { $type: 'string' } }
    ]
  }).toArray();

  console.log(results);
  client.close();
}

findByType();
```

Python

```
from pymongo import MongoClient

def find_by_type():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['documents']

    results = list(collection.find({
        '$or': [
            {'age': {'$type': 'number'}},
            {'email': {'$type': 'string'}},
            {'name': {'$type': 'string'}}
        ]
    }))

    print(results)
    client.close()

find_by_type()
```

Operatoren aktualisieren

Dieser Abschnitt enthält detaillierte Informationen zu Aktualisierungsoperatoren, die von Amazon DocumentDB unterstützt werden.

Themen

- $
- $[]
- $ [<Kennung>]
- $ hinzufügen ToSet
- $bit
- $currentDate
- $each
- $inc
- $max
- $min
- $mul
- $pop
- $position
- $pull
- $pullAll
- $push
- $rename
- $set
- $ gesetzt OnInsert
- $slice
- $sort
- $unset

$

Der $ Positionsoperator aktualisiert das erste Array-Element, das der Abfragebedingung entspricht. Er dient als Platzhalter für die Position des übereinstimmenden Array-Elements.

Parameter

- `field.$`: Das Array-Feld mit dem Positionsoperator zur Aktualisierung des ersten passenden Elements.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der $ Positionsoperator verwendet wird, um ein bestimmtes Array-Element zu aktualisieren.

Erstellen Sie Beispieldokumente

```
db.inventory.insertMany([
  { _id: 1, item: "Widget", quantities: [10, 20, 30] },
  { _id: 2, item: "Gadget", quantities: [5, 15, 25] }
]);
```

Beispiel für eine Abfrage

```
db.inventory.updateOne(
  { _id: 1, quantities: 20 },
  { $set: { "quantities.$": 22 } }
);
```

Ausgabe

```
{
  "_id" : 1,
  "item" : "Widget",
  "quantities" : [ 10, 22, 30 ]
}
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des $ Positionsoperators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function updateDocument() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('inventory');

  await collection.updateOne(
    { _id: 1, quantities: 20 },
    { $set: { "quantities.$": 22 } }
  );

  const updatedDocument = await collection.findOne({ _id: 1 });
  console.log(updatedDocument);

  await client.close();
}

updateDocument();
```

Python

```
from pymongo import MongoClient

def update_document():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.inventory

    collection.update_one(
        {'_id': 1, 'quantities': 20},
        {'$set': {'quantities.$': 22}}
    )

    updated_document = collection.find_one({'_id': 1})
    print(updated_document)
```

```
    client.close()

update_document()
```

$[]

Der `$[]` All-Position-Operator aktualisiert alle Elemente in einem Array. Er wird verwendet, wenn Sie jedes Element in einem Array-Feld ändern müssen.

Parameter

- `field.$[]`: Das Array-Feld mit dem All-Position-Operator zur Aktualisierung aller Elemente.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$[]` Operator verwendet wird, um alle Array-Elemente zu aktualisieren.

Erstellen Sie Beispieldokumente

```
db.products.insertOne({
  _id: 1,
  name: "Laptop",
  prices: [1000, 1100, 1200]
});
```

Beispiel für eine Abfrage

```
db.products.updateOne(
  { _id: 1 },
  { $inc: { "prices.$[]": 50 } }
);
```

Ausgabe

```
{
  "_id" : 1,
  "name" : "Laptop",
  "prices" : [ 1050, 1150, 1250 ]
}
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$[]` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function updateDocument() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('products');

  await collection.updateOne(
    { _id: 1 },
    { $inc: { "prices.$[]": 50 } }
  );

  const updatedDocument = await collection.findOne({ _id: 1 });
  console.log(updatedDocument);

  await client.close();
}

updateDocument();
```

Python

```
from pymongo import MongoClient

def update_document():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.products

    collection.update_one(
        {'_id': 1},
        {'$inc': {'prices.$[]': 50}}
```

```
    )

    updated_document = collection.find_one({'_id': 1})
    print(updated_document)

    client.close()

update_document()
```

$ [<Kennung>]

Der `$[<identifier>]` gefilterte Positionsoperator aktualisiert alle Array-Elemente, die den angegebenen Filterbedingungen entsprechen. Er wird mit der `arrayFilters` Option verwendet, Array-Elemente selektiv zu aktualisieren.

Parameter

- `field.$[identifier]`: Das Array-Feld mit dem gefilterten Positionsoperator.
- `arrayFilters`: Eine Reihe von Filterbedingungen, die bestimmen, welche Elemente aktualisiert werden sollen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$[<identifier>]` Operator verwendet wird, um bestimmte Array-Elemente auf der Grundlage einer Bedingung zu aktualisieren.

Erstellen Sie Beispieldokumente

```
db.students.insertOne({
  _id: 1,
  name: "Alice",
  grades: [
    { subject: "Math", score: 85 },
    { subject: "Science", score: 92 },
    { subject: "History", score: 78 }
  ]
});
```

Beispiel für eine Abfrage

```
db.students.updateOne(
  { _id: 1 },
  { $inc: { "grades.$[elem].score": 5 } },
  { arrayFilters: [{ "elem.score": { $gte: 80 } }] }
);
```

Ausgabe

```
{
  "_id" : 1,
  "name" : "Alice",
  "grades" : [
    { "subject" : "Math", "score" : 90 },
    { "subject" : "Science", "score" : 97 },
    { "subject" : "History", "score" : 78 }
  ]
}
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$[<identifier>]` Operators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function updateDocument() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('students');

  await collection.updateOne(
    { _id: 1 },
    { $inc: { "grades.$[elem].score": 5 } },
    { arrayFilters: [{ "elem.score": { $gte: 80 } }] }
  );

  const updatedDocument = await collection.findOne({ _id: 1 });
  console.log(updatedDocument);
```

```
  await client.close();
}

updateDocument();
```

Python

```
from pymongo import MongoClient

def update_document():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.students

    collection.update_one(
        {'_id': 1},
        {'$inc': {'grades.$[elem].score': 5}},
        array_filters=[{'elem.score': {'$gte': 80}}]
    )

    updated_document = collection.find_one({'_id': 1})
    print(updated_document)

    client.close()

update_document()
```

$ hinzufügen ToSet

Der `$addToSet` Operator in Amazon DocumentDB wird nur verwendet, um einem Array einen Wert hinzuzufügen, wenn der Wert nicht bereits im Array vorhanden ist. Dies ist nützlich, um sicherzustellen, dass ein Array eindeutige Elemente enthält.

Parameter

- `field`: Das zu aktualisierende Feld.
- `value`: Der Wert, der dem Array-Feld hinzugefügt werden soll. Dies kann ein einzelner Wert oder ein Ausdruck sein.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$addToSet` Operator verwendet wird, um einem Array eindeutige Elemente hinzuzufügen.

Erstellen Sie Beispieldokumente

```
db.products.insertMany([
  { "_id": 1, "item": "apple", "tags": ["fruit", "red", "round"] },
  { "_id": 2, "item": "banana", "tags": ["fruit", "yellow"] },
  { "_id": 3, "item": "cherry", "tags": ["fruit", "red"] }
])
```

Beispiel für eine Abfrage

```
db.products.update(
  { "item": "apple" },
  { $addToSet: { "tags": "green" } }
)
```

Ausgabe

```
{ "_id": 1, "item": "apple", "tags": ["fruit", "red", "round", "green"] }
```

In diesem Beispiel fügt der `$addToSet` Operator das Tag „green“ zum Array „tags“ des Dokuments hinzu, wobei das Feld „item“ den Wert „apple“ hat. Da „grün“ noch nicht im Array enthalten war, wurde es hinzugefügt.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$addToSet` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
```

```
  const db = client.db('test');
  const collection = db.collection('products');

  await collection.updateOne(
    { "item": "apple" },
    { $addToSet: { "tags": "green" } }
  );

  const updatedDoc = await collection.findOne({ "item": "apple" });
  console.log(updatedDoc);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.products

    collection.update_one(
        {"item": "apple"},
        {"$addToSet": {"tags": "green"}}
    )

    updated_doc = collection.find_one({"item": "apple"})
    print(updated_doc)

    client.close()

example()
```

$bit

Der `$bit` Operator in Amazon DocumentDB ermöglicht es Ihnen, bitweise Operationen an den Bits eines bestimmten Feldes durchzuführen. Dies kann für Aufgaben wie das Setzen, Löschen oder Überprüfen des Zustands einzelner Bits innerhalb einer Zahl nützlich sein.

Parameter

- `field`: Das Feld, in dem bitweise Operationen ausgeführt werden sollen.
- `and`: Ein ganzzahliger Wert, der verwendet wird, um eine bitweise UND-Operation für das Feld durchzuführen.
- `or`: Ein ganzzahliger Wert, der verwendet wird, um eine bitweise OR-Operation für das Feld durchzuführen.
- `xor`: Ein ganzzahliger Wert, der verwendet wird, um eine bitweise XOR-Operation für das Feld durchzuführen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$bit` Operator verwendet wird, um bitweise Operationen an einem numerischen Feld durchzuführen.

Erstellen Sie Beispieldokumente

```
db.numbers.insert([
  { "_id": 1, "number": 5 },
  { "_id": 2, "number": 12 }
])
```

Beispiel für eine Abfrage

```
db.numbers.update(
  { "_id": 1 },
  { "$bit": { "number": { "and": 3 } } }
)
```

Ausgabe

```
{
```

```
  "_id": 1,
  "number": 1
}
```

In diesem Beispiel wird der `$bit` Operator verwendet, um eine bitweise UND-Operation für das Feld „Zahl“ des Dokuments mit dem `_id` Wert 1 durchzuführen. Das Ergebnis ist, dass der Wert des Felds „Zahl“ auf 1 gesetzt wird, was das Ergebnis der bitweisen UND-Operation zwischen dem ursprünglichen Wert 5 und dem Wert 3 ist.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$bit` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('numbers');

  await collection.updateOne(
    { "_id": 1 },
    { "$bit": { "number": { "and": 3 } } }
  );

  const result = await collection.findOne({ "_id": 1 });
  console.log(result);

  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient
```

```
client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
db = client['test']
collection = db['numbers']

collection.update_one(
    {"_id": 1},
    {"$bit": {"number": {"and": 3}}}
)

result = collection.find_one({"_id": 1})
print(result)

client.close()
```

$currentDate

Der `$currentDate` Operator wird verwendet, um den Wert eines Felds auf das aktuelle Datum und die aktuelle Uhrzeit festzulegen. Dieser Operator ist nützlich, um ein Feld automatisch mit dem aktuellen Zeitstempel zu aktualisieren, wenn ein Dokument eingefügt oder aktualisiert wird.

Parameter

- `field`: Das Feld, das mit dem aktuellen Datum und der aktuellen Uhrzeit aktualisiert werden soll.
- `type`: (optional) Gibt den BSON-Typ an, der für das aktuelle Datum verwendet werden soll. Kann `date` oder `timestamp` sein.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$currentDate` Operator verwendet wird, um das `lastModified` Feld auf das aktuelle Datum und die aktuelle Uhrzeit einzustellen, wenn ein neues Dokument eingefügt wird.

Erstellen Sie Beispieldokumente

```
db.users.insert({
  name: "John Doe",
  email: "john.doe@example.com"
})
```

Beispiel für eine Abfrage

```
db.users.updateOne(
  { name: "John Doe" },
  { $currentDate: { lastModified: true } }
)
```

Aktualisiertes Dokument anzeigen

```
db.users.findOne({ name: "John Doe" })
```

Ausgabe

```
{
  _id: ObjectId('...'),
  name: 'John Doe',
  email: 'john.doe@example.com',
  lastModified: ISODate('2025-10-25T22:50:29.963Z')
}
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$currentDate` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function updateUserWithCurrentDate() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const users = db.collection('users');

  await users.updateOne(
    { name: 'John Doe' },
    { $currentDate: { lastModified: true } }
  );

  console.log('User updated with current date');
```

```
  client.close();
}

updateUserWithCurrentDate();
```

Python

```
from pymongo import MongoClient

def update_user_with_current_date():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    users = db.users

    result = users.update_one(
        {'name': 'John Doe'},
        {'$currentDate': {'lastModified': True}}
    )

    print('User updated with current date')
    client.close()

update_user_with_current_date()
```

$each

Der `$each` Operator wird in Verbindung mit anderen Aktualisierungsoperatoren wie `$push` und verwendet`$addToSet`, um einem Array-Feld mehrere Werte hinzuzufügen. Es ermöglicht das Hinzufügen mehrerer Elemente zu einem Array in einem einzigen Vorgang, anstatt mehrere Aktualisierungsvorgänge ausführen zu müssen.

Parameter

- `value`: Das Array von Werten, die dem Array-Feld hinzugefügt werden sollen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$each` Operator zusammen mit dem `$push` Operator verwendet wird, um einem Array-Feld mehrere Elemente hinzuzufügen.

Erstellen Sie Beispieldokumente

```
db.fruits.insertOne({
  _id: 1,
  fruits: ["apple", "banana"]
})
```

Beispiel für eine Abfrage

```
db.fruits.updateOne(
  { _id: 1 },
  { $push: { fruits: { $each: ["cherry", "durian", "elderberry"] } } }
)
```

Aktualisiertes Dokument anzeigen

```
db.fruits.findOne({ _id: 1 })
```

Ausgabe

```
{
  _id: 1,
  fruits: [ 'apple', 'banana', 'cherry', 'durian', 'elderberry' ]
}
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$each` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('fruits');

  await collection.updateOne(
```

```
    { _id: 1 },
    { $push: { fruits: { $each: ["cherry", "durian", "elderberry"] } } }
  );

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['fruits']

    collection.update_one(
        {'_id': 1},
        {'$push': {'fruits': {'$each': ['cherry', 'durian', 'elderberry']}}}
    )

    client.close()

example()
```

$inc

Der `$inc` Operator wird verwendet, um den Wert eines Felds um einen bestimmten Betrag zu erhöhen. Er wird verwendet, um ein numerisches Feld, z. B. einen Zähler oder eine Bewertung, zu aktualisieren, ohne dass der aktuelle Wert abgerufen, der neue Wert berechnet und das Feld anschließend aktualisiert werden muss.

Parameter

- `field`: Der Name des Feldes, das erhöht werden soll.
- `amount`: Der Betrag, um den das Feld erhöht werden soll. Dies kann ein positiver oder negativer Wert sein.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$inc` Operator verwendet wird, um das `age` Feld eines Dokuments zu erhöhen.

Erstellen Sie Beispieldokumente

```
db.users.insertOne({_id: 123, name: "John Doe", age: 30})
```

Beispiel für eine Abfrage

```
db.users.updateOne({_id: 123}, {$inc: {age: 1}})
```

Aktualisiertes Dokument anzeigen

```
db.users.findOne({_id: 123})
```

Ausgabe

```
{ "_id" : 123, "name" : "John Doe", "age" : 31 }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$inc` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function updateWithInc() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('users');

  const result = await collection.updateOne(
    { _id: 123 },
    { $inc: { age: 1 } }
```

```
  );

  console.log(result);

  await client.close();
}

updateWithInc();
```

Python

```
from pymongo import MongoClient

def update_with_inc():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['users']

    result = collection.update_one(
        {'_id': 123},
        {'$inc': {'age': 1}}
    )

    print(result.modified_count)

    client.close()

update_with_inc()
```

$max

Der `$max` Aktualisierungsoperator aktualisiert den Wert eines Felds nur, wenn der angegebene Wert größer als der aktuelle Feldwert ist. Dieser Operator ist nützlich, um Maximalwerte bei Aktualisierungen beizubehalten.

Parameter

- `field`: Das zu aktualisierende Feld.
- `value`: Der Wert, der mit dem aktuellen Feldwert verglichen werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$max` Operator verwendet wird, um die höchste aufgezeichnete Punktzahl für einen Spieler zu aktualisieren.

Erstellen Sie Beispieldokumente

```
db.scores.insertMany([
  { _id: 1, player: "Alice", highScore: 85 },
  { _id: 2, player: "Bob", highScore: 92 },
  { _id: 3, player: "Charlie", highScore: 78 }
])
```

Beispiel aktualisieren

```
db.scores.updateOne(
  { _id: 1 },
  { $max: { highScore: 95 } }
)
```

Ergebnis

Das `highScore` Feld für Alice wird auf 95 aktualisiert, weil 95 größer als der aktuelle Wert von 85 ist.

```
{ "_id": 1, "player": "Alice", "highScore": 95 }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$max` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('scores');
```

```
  const result = await collection.updateOne(
    { _id: 1 },
    { $max: { highScore: 95 } }
  );

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['scores']

    result = collection.update_one(
        { '_id': 1 },
        { '$max': { 'highScore': 95 } }
    )

    print(result)
    client.close()

example()
```

$min

Der `$min` Aktualisierungsoperator aktualisiert den Wert eines Felds nur, wenn der angegebene Wert kleiner als der aktuelle Feldwert ist. Dieser Operator ist nützlich, um Mindestwerte bei Aktualisierungen beizubehalten.

Parameter

- `field`: Das zu aktualisierende Feld.

- `value`: Der Wert, der mit dem aktuellen Feldwert verglichen werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$min` Operator verwendet wird, um die niedrigste gemessene Temperatur für eine Wetterstation zu aktualisieren.

Erstellen Sie Beispieldokumente

```
db.weather.insertMany([
  { _id: 1, station: "Station A", lowestTemp: 15 },
  { _id: 2, station: "Station B", lowestTemp: 20 },
  { _id: 3, station: "Station C", lowestTemp: 18 }
])
```

Beispiel aktualisieren

```
db.weather.updateOne(
  { _id: 1 },
  { $min: { lowestTemp: 12 } }
)
```

Ergebnis

Das `lowestTemp` Feld für Station A wird auf 12 aktualisiert, weil 12 kleiner als der aktuelle Wert von 15 ist.

```
{ "_id": 1, "station": "Station A", "lowestTemp": 12 }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$min` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
```

```
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('weather');

  const result = await collection.updateOne(
    { _id: 1 },
    { $min: { lowestTemp: 12 } }
  );

  console.log(result);
  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['weather']

    result = collection.update_one(
        { '_id': 1 },
        { '$min': { 'lowestTemp': 12 } }
    )

    print(result)
    client.close()

example()
```

$mul

Der `$mul` Operator in Amazon DocumentDB wird verwendet, um den Wert eines Felds mit einer bestimmten Zahl zu multiplizieren. Dies kann nützlich sein, um mehrere Dokumente automatisch

und konsistent zu aktualisieren, z. B. um Flugmeilen auf der Grundlage eines Kreditkartenstatus zu aktualisieren.

Parameter

- `field`: Das Feld, das multipliziert werden soll.
- `multiplier`: Die Zahl, mit der der Feldwert multipliziert werden soll.

Beispiel (MongoDB Shell)

Dieses Beispiel zeigt, wie der `$mul` Operator verwendet wird, um den `flight_miles` Wert für alle Dokumente zu verdoppeln, in denen sich das `credit_card` Feld befindet`true`.

Erstellen Sie Beispieldokumente

```
db.miles.insertMany([
  { "_id": 1, "member_since": new Date("1987-01-01"), "credit_card": false,
 "flight_miles": [1205, 2560, 880] },
  { "_id": 2, "member_since": new Date("1982-01-01"), "credit_card": true,
 "flight_miles": [2410, 5120, 1780, 5560] },
  { "_id": 3, "member_since": new Date("1999-01-01"), "credit_card": true,
 "flight_miles": [2410, 1760] }
]);
```

Beispiel für eine Abfrage

```
db.miles.update(
  { "credit_card": { "$eq": true } },
  { "$mul": { "flight_miles.$[]": NumberInt(2) } },
  { "multi": true }
);
```

Ausgabe

```
{ "_id" : 1, "member_since" : ISODate("1987-01-01T00:00:00Z"), "credit_card" : false,
 "flight_miles" : [ 1205, 2560, 880 ] }
{ "_id" : 2, "member_since" : ISODate("1982-01-01T00:00:00Z"), "credit_card" : true,
 "flight_miles" : [ 4820, 10240, 3560, 11120 ] }
{ "_id" : 3, "member_since" : ISODate("1999-01-01T00:00:00Z"), "credit_card" : true,
 "flight_miles" : [ 4820, 3520 ] }
```

Für Kunden, die eine Kreditkarte besitzen, wurden ihre Flugmeilen verdoppelt.

Der `$[]` Positions-Array-Operator wird verwendet, um die `$mul` Operation auf jedes Element im `flight_miles` Array anzuwenden.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$mul` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function updateFlightMiles() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('miles');

  await collection.updateMany(
    { credit_card: true },
    { $mul: { 'flight_miles.$[]': 2 } }
  );

  const documents = await collection.find().toArray();
  console.log(documents);

  await client.close();
}

updateFlightMiles();
```

Python

```
from pymongo import MongoClient

def update_flight_miles():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
```

```
    collection = db.miles

    collection.update_many(
        {'credit_card': True},
        {'$mul': {'flight_miles.$[]': 2}}
    )

    documents = list(collection.find())
    print(documents)

    client.close()

update_flight_miles()
```

$pop

Der $pop Operator in Amazon DocumentDB wird verwendet, um das erste oder letzte Element aus einem Array-Feld zu entfernen. Dies ist besonders nützlich, wenn Sie ein Array mit fester Größe verwalten oder eine warteschlangenartige Datenstruktur innerhalb eines Dokuments implementieren müssen.

Parameter

- `field`: Der Name des Array-Felds, aus dem ein Element entfernt werden soll.
- `value`: Ein ganzzahliger Wert, der die Position des zu entfernenden Elements bestimmt. Ein Wert von 1 entfernt das letzte Element, während ein Wert von das erste Element `-1` entfernt.

Beispiel (MongoDB Shell)

Dieses Beispiel zeigt, wie der $pop Operator verwendet wird, um das erste und das letzte Element aus einem Array-Feld zu entfernen.

Erstellen Sie Beispieldokumente

```
db.users.insertMany([
  { "_id": 1, "name": "John Doe", "hobbies": ["reading", "swimming", "hiking"] },
  { "_id": 2, "name": "Jane Smith", "hobbies": ["cooking", "gardening", "painting"] }
])
```

Beispiel für eine Abfrage

```
// Remove the first element from the "hobbies" array
db.users.update({ "_id": 1 }, { $pop: { "hobbies": -1 } })

// Remove the last element from the "hobbies" array
db.users.update({ "_id": 2 }, { $pop: { "hobbies": 1 } })
```

Ausgabe

```
{ "_id" : 1, "name" : "John Doe", "hobbies" : [ "swimming", "hiking" ] }
{ "_id" : 2, "name" : "Jane Smith", "hobbies" : [ "cooking", "gardening" ] }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des $pop Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('users');

  // Remove the first element from the "hobbies" array
  await collection.updateOne({ "_id": 1 }, { $pop: { "hobbies": -1 } });

  // Remove the last element from the "hobbies" array
  await collection.updateOne({ "_id": 2 }, { $pop: { "hobbies": 1 } });

  const users = await collection.find().toArray();
  console.log(users);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['users']

    # Remove the first element from the "hobbies" array
    collection.update_one({"_id": 1}, {"$pop": {"hobbies": -1}})

    # Remove the last element from the "hobbies" array
    collection.update_one({"_id": 2}, {"$pop": {"hobbies": 1}})

    users = list(collection.find())
    print(users)

    client.close()

example()
```

$position

Der `$position` Modifikator in Amazon DocumentDB gibt die Position im Array an, an der der `$push` Operator Elemente einfügt. Ohne den `$position` Modifikator fügt der `$push` Operator Elemente am Ende des Arrays ein.

Parameter

- `field`: Das zu aktualisierende Array-Feld.
- num: Die Position im Array, an der Elemente eingefügt werden sollen, basierend auf einer Nullindizierung.

Hinweis: Um den `$position` Modifikator verwenden zu können, muss er zusammen mit dem Modifikator erscheinen. `$each`

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$position` Operator verwendet wird, um Aufgaben an bestimmten Positionen in einem Projektmanagementsystem einzufügen.

Erstellen Sie Beispieldokumente

```
db.projects.insertOne({ "_id": 1, "name": "Website Redesign", "tasks": ["Design
 mockups"] })
```

Abfragebeispiel 1 — Fügen Sie am Anfang dringende Aufgaben hinzu

```
db.projects.updateOne(
   { _id: 1 },
   {
     $push: {
        tasks: {
           $each: ["Security audit", "Performance review"],
           $position: 0
        }
     }
   }
)
```

Ausgang 1

```
{ "_id": 1, "name": "Website Redesign", "tasks": ["Security audit", "Performance
 review", "Design mockups"] }
```

Abfragebeispiel 2 — Aufgaben an einer bestimmten Position hinzufügen

```
db.projects.insertOne({ "_id": 2, "name": "Mobile App", "tasks": ["Setup project",
 "Create wireframes", "Deploy to store"] })

db.projects.updateOne(
   { _id: 2 },
   {
     $push: {
        tasks: {
           $each: ["Code review", "Testing phase"],
           $position: 2
```

```
        }
      }
    }
)
```

Ausgabe 2

```
{ "_id": 2, "name": "Mobile App", "tasks": ["Setup project", "Create wireframes", "Code
 review", "Testing phase", "Deploy to store"] }
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$position` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function insertTasksAtPosition() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('projects');

  await collection.updateOne(
    { _id: 1 },
    {
      $push: {
        tasks: {
          $each: ["Security audit", "Performance review"],
          $position: 0
        }
      }
    }
  );

  const updatedProject = await collection.findOne({ _id: 1 });
  console.log(updatedProject);

  await client.close();
}
```

```
insertTasksAtPosition();
```

Python

```
from pymongo import MongoClient

def insert_tasks_at_position():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['projects']

    result = collection.update_one(
        {'_id': 1},
        {
            '$push': {
                'tasks': {
                    '$each': ['Security audit', 'Performance review'],
                    '$position': 0
                }
            }
        }
    )

    updated_project = collection.find_one({'_id': 1})
    print(updated_project)

    client.close()

insert_tasks_at_position()
```

$pull

Der `$pull` Operator wird verwendet, um alle Instanzen eines Werts oder von Werten, die einer bestimmten Bedingung entsprechen, aus einem Array zu entfernen. Dieser Operator ist nützlich, wenn Sie bestimmte Elemente aus einem Array-Feld innerhalb eines Dokuments entfernen müssen.

Parameter

- `field`: Der Name des Array-Feldes, aus dem die Werte entfernt werden sollen.

- `value`: Der Wert oder die Bedingung, die bestimmt, welche Elemente aus dem Array entfernt werden sollen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$pull` Operator verwendet wird, um Elemente aus einem Array-Feld zu entfernen.

Erstellen Sie Beispieldokumente

```
db.restaurants.insertMany([
  {
    name: "Pizza Hut",
    cuisine: "Italian",
    features: ["Delivery", "Takeout", "Dine-in"]
  },
  {
    name: "Sushi Saito",
    cuisine: "Japanese",
    features: ["Dine-in", "Private Dining"]
  },
  {
    name: "Taco Bell",
    cuisine: "Mexican",
    features: ["Delivery", "Takeout", "Drive-thru"]
  }
])
```

Beispiel für eine Abfrage

```
db.restaurants.updateMany(
  { cuisine: "Italian" },
  { $pull: { features: "Takeout" } }
)
```

Ausgabe

```
{
  "acknowledged" : true,
  "matchedCount" : 1,
  "modifiedCount" : 1
```

```
}
```

Die obige Abfrage entfernt die Funktion „Mitnehmen“ aus allen Dokumenten, in denen das `cuisine` Feld „Italienisch“ ist.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$pull` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function updateRestaurants() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const restaurants = db.collection('restaurants');

  await restaurants.updateMany(
    { cuisine: 'Italian' },
    { $pull: { features: 'Takeout' } }
  );

  const updatedRestaurants = await restaurants.find({}).toArray();
  console.log(updatedRestaurants);

  await client.close();
}

updateRestaurants();
```

Python

```
from pymongo import MongoClient

def update_restaurants():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
```

```
    restaurants = db.restaurants

    result = restaurants.update_many(
        { 'cuisine': 'Italian' },
        { '$pull': { 'features': 'Takeout' } }
    )

    updated_restaurants = list(restaurants.find({}))
    print(updated_restaurants)

    client.close()

update_restaurants()
```

$pullAll

Der `$pullAll` Operator in Amazon DocumentDB wird verwendet, um alle Instanzen der angegebenen Werte aus einem Array-Feld zu entfernen. Dies ist besonders nützlich, wenn Sie in einem einzigen Vorgang mehrere Elemente aus einem Array entfernen müssen.

Parameter

- `field`: Der Name des Array-Feldes, aus dem die Elemente entfernt werden sollen.
- `value`: Eine Reihe von Werten, die aus dem Array-Feld entfernt werden sollen.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$pullAll` Operator verwendet wird, um mehrere Elemente aus einem Array-Feld zu entfernen.

Erstellen Sie Beispieldokumente

```
db.restaurants.insert([
  {
    "name": "Taj Mahal",
    "cuisine": "Indian",
    "features": ["Private Dining", "Live Music"]
  },
  {
    "name": "Golden Palace",
    "cuisine": "Chinese",
```

```
    "features": ["Private Dining", "Takeout"]
  },
  {
    "name": "Olive Garden",
    "cuisine": "Italian",
    "features": ["Private Dining", "Outdoor Seating"]
  }
])
```

Beispiel für eine Abfrage

```
db.restaurants.update(
  { "name": "Taj Mahal" },
  { $pullAll: { "features": ["Private Dining", "Live Music"] } }
)
```

Ausgabe

```
{
  "name": "Taj Mahal",
  "cuisine": "Indian",
  "features": []
}
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$pullAll` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function main() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('restaurants');

  await collection.updateMany(
    { "name": "Taj Mahal" },
```

```
    { $pullAll: { "features": ["Private Dining", "Live Music"] } }
  );

  const updatedDocument = await collection.findOne({ "name": "Taj Mahal" });
  console.log(updatedDocument);

  await client.close();
}

main();
```

Python

```
from pymongo import MongoClient

def main():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['restaurants']

    collection.update_many(
        {"name": "Taj Mahal"},
        {"$pullAll": {"features": ["Private Dining", "Live Music"]}}
    )

    updated_document = collection.find_one({"name": "Taj Mahal"})
    print(updated_document)

    client.close()

if __name__ == '__main__':
    main()
```

$push

Der `$push` Operator in Amazon DocumentDB wird verwendet, um ein Element zu einem Array-Feld in einem Dokument hinzuzufügen. Dieser Operator ist besonders nützlich, wenn Sie neue Daten an ein vorhandenes Array anhängen müssen, ohne das gesamte Array zu überschreiben.

Parameter

- `field`: Der Name des Array-Feldes, zu dem das neue Element hinzugefügt werden soll.
- `value`: Der Wert, der dem Array hinzugefügt werden soll.
- `position`: (optional) Ein Modifikator, der die Position im Array angibt, an der das neue Element hinzugefügt werden soll. Zu den unterstützten Modifikatoren gehören $ (am Ende des Arrays hinzufügen) und `$[]` (am Ende des Arrays hinzufügen, dabei alle Array-Filter ignorieren).

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$push` Operator verwendet wird, um einem Array-Feld in einem Dokument neue Elemente hinzuzufügen.

Erstellen Sie Beispieldokumente

```
db.users.insert([
  { _id: 1, name: "John Doe", hobbies: ["reading", "swimming"] },
  { _id: 2, name: "Jane Smith", hobbies: ["gardening", "cooking"] }
])
```

Beispiel für eine Abfrage

```
db.users.updateOne(
  { _id: 1 },
  { $push: { hobbies: "hiking" } }
)
```

Ausgabe

```
{
  "acknowledged" : true,
  "matchedCount" : 1,
  "modifiedCount" : 1
}
```

Nach dem Ausführen des Updates `_id: 1` wird das `hobbies` Array für das Dokument mit aktualisiert`["reading", "swimming", "hiking"]`.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$push` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function updateDocument() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('users');

  const result = await collection.updateOne(
    { _id: 1 },
    { $push: { hobbies: "hiking" } }
  );

  console.log(result);
  client.close();
}

updateDocument();
```

Python

```
from pymongo import MongoClient

def update_document():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['users']

    result = collection.update_one(
        {'_id': 1},
        {'$push': {'hobbies': 'hiking'}}
    )

    print(result.raw_result)
    client.close()

update_document()
```

$rename

Der `$rename` Operator in Amazon DocumentDB wird verwendet, um ein Feld in einem Dokument umzubenennen. Dieser Operator kann besonders nützlich sein, wenn Sie die Struktur Ihrer Dokumente aktualisieren oder sie an neue Datenmodelle anpassen müssen.

Parameter

- `field`: Das Feld, das umbenannt werden soll.
- `newName`: Der neue Name für das Feld.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie Sie den `$rename` Operator verwenden, um das `"Date.DoW"` Feld `"Date.DayOfWeek"` in einem Dokument umzubenennen, in dem das `"DocName"` Feld auf `"Document 1"` gesetzt ist.

Erstellen Sie Beispieldokumente

```
db.example.insertOne({
    "DocName": "Document 1",
    "Date": {
        "Month": 4,
        "Day": 18,
        "Year": 1987,
        "DoW": "Saturday"
    },
    "Words": 2482
})
```

Beispiel für eine Abfrage

```
db.example.update(
    { "DocName": "Document 1" },
    { $rename: { "Date.DoW": "Date.DayOfWeek" } }
)
```

Ausgabe

```
{
```

```
    "DocName": "Document 1",
    "Date": {
        "Month": 4,
        "Day": 18,
        "Year": 1987,
        "DayOfWeek": "Saturday"
    },
    "Words": 2482
}
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$rename` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
    const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
    const db = client.db('test');
    const collection = db.collection('example');

    await collection.updateOne(
        { "DocName": "Document 1" },
        { $rename: { "Date.DoW": "Date.DayOfWeek" } }
    );

    const updatedDoc = await collection.findOne({ "DocName": "Document 1" });
    console.log(updatedDoc);

    await client.close();
}

example();
```

Python

```
from pymongo import MongoClient
```

```
def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['example']

    collection.update_one(
        {"DocName": "Document 1"},
        {"$rename": {"Date.DoW": "Date.DayOfWeek"}}
    )

    updated_doc = collection.find_one({"DocName": "Document 1"})
    print(updated_doc)

    client.close()

example()
```

$set

Der `$set` Operator in Amazon DocumentDB wird verwendet, um den Wert eines bestimmten Felds in einem Dokument zu aktualisieren. Mit diesem Operator können Sie neue Felder hinzufügen oder bestehende Felder innerhalb eines Dokuments ändern. Es ist ein grundlegender Aktualisierungsoperator im MongoDB-Java-Treiber, der mit Amazon DocumentDB kompatibel ist.

Parameter

- `field`: Das zu aktualisierende Feld.
- `value`: Der neue Wert für das Feld.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$set` Operator verwendet wird, um das `Item` Feld in einem Dokument zu aktualisieren.

Erstellen Sie Beispieldokumente

```
db.example.insert([
  {
    "Item": "Pen",
```

```
    "Colors": ["Red", "Green", "Blue", "Black"],
    "Inventory": {
      "OnHand": 244,
      "MinOnHand": 72
    }
  },
  {
    "Item": "Poster Paint",
    "Colors": ["Red", "Green", "Blue", "White"],
    "Inventory": {
      "OnHand": 120,
      "MinOnHand": 36
    }
  }
])
```

Beispiel für eine Abfrage

```
db.example.update(
  { "Item": "Pen" },
  { $set: { "Item": "Gel Pen" } }
)
```

Ausgabe

```
{
  "Item": "Gel Pen",
  "Colors": ["Red", "Green", "Blue", "Black"],
  "Inventory": {
    "OnHand": 244,
    "MinOnHand": 72
  }
}
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$set` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');
```

```
async function updateDocument() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('example');

  await collection.updateOne(
    { "Item": "Pen" },
    { $set: { "Item": "Gel Pen" } }
  );

  const updatedDocument = await collection.findOne({ "Item": "Gel Pen" });
  console.log(updatedDocument);

  await client.close();
}

updateDocument();
```

Python

```
from pymongo import MongoClient

def update_document():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.example

    collection.update_one(
        {"Item": "Pen"},
        {"$set": {"Item": "Gel Pen"}}
    )

    updated_document = collection.find_one({"Item": "Gel Pen"})
    print(updated_document)

    client.close()

update_document()
```

$ gesetzt OnInsert

Der `$setOnInsert` Operator in Amazon DocumentDB wird verwendet, um den Wert eines Felds festzulegen, wenn ein Dokument eingefügt wird. Er hat jedoch keine Auswirkung, wenn das Dokument aktualisiert wird.

Parameter

- `field`: Das Feld, das festgelegt werden soll.
- `value`: Der Wert, der dem Feld zugewiesen werden soll.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt die Verwendung des `$setOnInsert` Operators in Amazon DocumentDB. Es erstellt ein neues Dokument, wenn das Dokument noch nicht existiert, hat aber keine Auswirkung, wenn das Dokument aktualisiert wird.

Erstellen Sie Beispieldokumente

```
db.users.insertOne({
  _id: 1,
  name: "John Doe",
  age: 30
})
```

Abfragebeispiel 1 — Bestehendes Dokument aktualisieren

```
db.users.update(
  { _id: 1 },
  {
    $set: { age: 31 },
    $setOnInsert: { createdAt: new Date() }
  },
  { upsert: true }
)
```

Ausgabe 1

```
{
  _id: 1,
```

```
  name: "John Doe",
  age: 31
}
```

Die Ausgabe zeigt, dass das Dokument aktualisiert wurde, das `createdAt` Feld jedoch NICHT hinzugefügt wurde, da das Dokument bereits vorhanden war. Der `$setOnInsert` Operator gilt nur beim Einfügen neuer Dokumente.

Abfragebeispiel 2 — Neues Dokument einfügen (upsert)

```
db.users.update(
  { _id: 2 },
  {
    $set: { name: "Jane Smith", age: 25 },
    $setOnInsert: { createdAt: new Date() }
  },
  { upsert: true }
)
```

Ausgabe 2

```
{
  _id: 2,
  name: "Jane Smith",
  age: 25,
  createdAt: ISODate("2025-10-31T09:57:52.459Z")
}
}
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$setOnInsert` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function updateWithSetOnInsert() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
```

```
  const db = client.db('test');
  const users = db.collection('users');

  await users.updateOne(
    { _id: 1 },
    {
      $set: { age: 31 },
      $setOnInsert: { createdAt: new Date() }
    },
    { upsert: true }
  );

  const updatedUser = await users.findOne({ _id: 1 });
  console.log(updatedUser);

  await client.close();
}

updateWithSetOnInsert();
```

Python

```
from pymongo import MongoClient

def update_with_set_on_insert():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    users = db['users']

    result = users.update_one(
        {'_id': 1},
        {
            '$set': {'age': 31},
            '$setOnInsert': {'createdAt': datetime.datetime.now()}
        },
        upsert=True
    )

    updated_user = users.find_one({'_id': 1})
    print(updated_user)
```

```
    client.close()

update_with_set_on_insert()
```

$slice

Der `$slice` Aktualisierungsoperator ändert ein Array, indem er seine Größe begrenzt. Wenn er zusammen mit dem `$push` Operator verwendet wird, schränkt er die Anzahl der Elemente in einem Array ein und behält nur die angegebene Anzahl der neuesten oder ältesten Elemente bei.

Parameter

- `field`: Das zu ändernde Array-Feld.
- `count`: Maximale Anzahl von Elementen, die beibehalten werden sollen. Positive Werte behalten die ersten N Elemente, negative Werte behalten die letzten N Elemente.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$slice` Aktualisierungsoperator with verwendet wird`$push`, um ein Array mit fester Größe mit aktuellen Ergebnissen zu verwalten.

Erstellen Sie Beispieldokumente

```
db.students.insertOne({
  _id: 1,
  name: "Alice",
  scores: [85, 90, 78]
});
```

Beispiel für eine Abfrage

```
db.students.updateOne(
  { _id: 1 },
  {
    $push: {
      scores: {
        $each: [92, 88],
        $slice: -3
      }
    }
```

```
  }
)
```

Ausgabe

```
{
  "_id" : 1,
  "name" : "Alice",
  "scores" : [ 78, 92, 88 ]
}
```

In diesem Beispiel behält der `$slice: -3` Modifikator nur die letzten drei Elemente bei, nachdem er neue Werte in das Array übertragen hat.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$slice` Aktualisierungsoperators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function updateDocument() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('students');

  await collection.updateOne(
    { _id: 1 },
    {
      $push: {
        scores: {
          $each: [92, 88],
          $slice: -3
        }
      }
    }
  );

  const updatedDocument = await collection.findOne({ _id: 1 });
```

```
  console.log(updatedDocument);

  await client.close();
}

updateDocument();
```

Python

```
from pymongo import MongoClient

def update_document():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.students

    collection.update_one(
        {'_id': 1},
        {
            '$push': {
                'scores': {
                    '$each': [92, 88],
                    '$slice': -3
                }
            }
        }
    )

    updated_document = collection.find_one({'_id': 1})
    print(updated_document)

    client.close()

update_document()
```

$sort

Der `$sort` Aktualisierungsmodifikator ordnet Array-Elemente an, wenn er zusammen mit dem `$push` Operator verwendet wird. Er ordnet Array-Elemente auf der Grundlage der angegebenen Feldwerte oder der Elemente selbst in aufsteigender oder absteigender Reihenfolge an.

Parameter

- `field`: Das zu ändernde Array-Feld.
- `order`: Wird 1 für aufsteigende oder `-1` absteigende Reihenfolge verwendet.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$sort` Modifikator mit verwendet wird`$push`, um neue Quizergebnisse hinzuzufügen und sie in absteigender Reihenfolge zu sortieren.

Erstellen Sie Beispieldokumente

```
db.students.insertOne({
  _id: 1,
  name: "Bob",
  quizzes: [
    { score: 85, date: "2024-01-15" },
    { score: 92, date: "2024-02-10" }
  ]
});
```

Beispiel für eine Abfrage

```
db.students.updateOne(
  { _id: 1 },
  {
    $push: {
      quizzes: {
        $each: [{ score: 78, date: "2024-03-05" }],
        $sort: { score: -1 }
      }
    }
  }
)
```

Ausgabe

```
{
  "_id" : 1,
  "name" : "Bob",
```

```
  "quizzes" : [
    { "score" : 92, "date" : "2024-02-10" },
    { "score" : 85, "date" : "2024-01-15" },
    { "score" : 78, "date" : "2024-03-05" }
  ]
}
```

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$sort` Aktualisierungsmodifikators anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function updateDocument() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('students');

  await collection.updateOne(
    { _id: 1 },
    {
      $push: {
        quizzes: {
          $each: [{ score: 78, date: "2024-03-05" }],
          $sort: { score: -1 }
        }
      }
    }
  );

  const updatedDocument = await collection.findOne({ _id: 1 });
  console.log(updatedDocument);

  await client.close();
}

updateDocument();
```

Python

```
from pymongo import MongoClient

def update_document():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.students

    collection.update_one(
        {'_id': 1},
        {
            '$push': {
                'quizzes': {
                    '$each': [{'score': 78, 'date': '2024-03-05'}],
                    '$sort': {'score': -1}
                }
            }
        }
    )

    updated_document = collection.find_one({'_id': 1})
    print(updated_document)

    client.close()

update_document()
```

$unset

Der `$unset` Operator in Amazon DocumentDB wird verwendet, um ein bestimmtes Feld aus einem Dokument zu entfernen. Wenn ein Feld mit entfernt wird`$unset`, wird das Feld aus dem Dokument gelöscht, und die Dokumentgröße wird entsprechend reduziert. Dies kann nützlich sein, wenn Sie unnötige Daten aus Ihren Dokumenten entfernen möchten.

Parameter

- `field`: Das Feld, das aus dem Dokument entfernt werden soll. Dies kann ein einzelnes Feld oder ein gepunkteter Pfad zu einem verschachtelten Feld sein.

Beispiel (MongoDB Shell)

Das folgende Beispiel zeigt, wie der `$unset` Operator verwendet wird, um das `Words` Feld aus einem Dokument in der `example` Sammlung zu entfernen.

Erstellen Sie Beispieldokumente

```
db.example.insert({
    "DocName": "Document 1",
    "Date": {
        "Month": 4,
        "Day": 18,
        "Year": 1987,
        "DoW": "Saturday"
    },
    "Words": 2482
})
```

Beispiel für eine Abfrage

```
db.example.update(
    { "DocName" : "Document 1" },
    { $unset: { Words:1 } }
)
```

Ausgabe

```
{
    "DocName": "Document 1",
    "Date": {
        "Month": 4,
        "Day": 18,
        "Year": 1987,
        "DoW": "Saturday"
    }
}
```

In diesem Beispiel wird der `$unset` Operator verwendet, um das `Words` Feld aus dem Dokument zu entfernen, das „Dokument 1" `DocName` entspricht. Das resultierende Dokument enthält das `Words` Feld nicht mehr.

Codebeispiele

Um ein Codebeispiel für die Verwendung des `$unset` Befehls anzuzeigen, wählen Sie die Registerkarte für die Sprache, die Sie verwenden möchten:

Node.js

```
const { MongoClient } = require('mongodb');

async function removeField() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('example');

  const result = await collection.updateOne(
    { "DocName": "Document 1" },
    { $unset: { "Words": 1 } }
  );

  console.log(`Modified ${result.modifiedCount} document(s)`);
  client.close();
}

removeField();
```

Python

```
from pymongo import MongoClient

def remove_field():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['example']

    result = collection.update_one(
        {"DocName": "Document 1"},
        {"$unset": {"Words": 1}}
    )
```

```
    print(f"Modified {result.modified_count} document(s)")
    client.close()

remove_field()
```

Verwaltung von Amazon DocumentDB-Clustern und -Instances

In diesen Abschnitten werden die verschiedenen Komponenten und die damit verbundenen Aufgaben für die Verwaltung Ihrer Amazon DocumentDB-Implementierung (mit MongoDB-Kompatibilität) behandelt.

Themen

- Überblick über die betrieblichen Aufgaben von Amazon DocumentDB
- Überblick über die globalen Amazon DocumentDB-Cluster
- Verwaltung von Amazon DocumentDB-Clustern
- Verwaltung von Amazon DocumentDB DocumentDB-Instances
- Verwaltung von Amazon DocumentDB-Subnetzgruppen
- Amazon DocumentDB — Hochverfügbarkeit und Replikation
- Amazon DocumentDB DocumentDB-Ereignisse verwalten
- Verwaltung von Amazon DocumentDB-Cluster-Parametergruppen
- Grundlegendes zu Amazon DocumentDB DocumentDB-Endpunkten
- Grundlegendes zu Amazon DocumentDB Amazon Resource Names (ARNs)
- Taggen von Amazon DocumentDB DocumentDB-Ressourcen
- Wartung von Amazon DocumentDB
- Grundlegendes zu serviceverknüpften Rollen

Überblick über die betrieblichen Aufgaben von Amazon DocumentDB

In diesem Abschnitt werden Betriebsaufgaben für Ihren Amazon DocumentDB-Cluster behandelt und wie Sie diese Aufgaben mithilfe von ausführen. AWS CLI

Themen

- Hinzufügen einer Replik zu einem Amazon DocumentDB-Cluster
- Beschreibung von Clustern und Instanzen

- Einen Cluster-Snapshot erstellen
- Wiederherstellung aus einem Snapshot
- Eine Instanz aus einem Cluster entfernen
- Löschen eines Clusters

Hinzufügen einer Replik zu einem Amazon DocumentDB-Cluster

Nachdem Sie die primäre Instance für Ihren Amazon DocumentDB-Cluster erstellt haben, können Sie eine oder mehrere Repliken hinzufügen. Ein Replikat ist eine schreibgeschützte Instance, die zwei Zwecken dient:

- Skalierbarkeit — Wenn Sie eine große Anzahl von Clients haben, die gleichzeitigen Zugriff benötigen, können Sie weitere Replikate für die Leseskalierung hinzufügen.
- Hochverfügbarkeit — Wenn die primäre Instance ausfällt, führt Amazon DocumentDB automatisch ein Failover zu einer Replikat-Instance durch und bestimmt diese als neue primäre Instance. Wenn ein Replikat ausfällt, können andere Instances im Cluster nach wie vor für die Bearbeitung von Anfragen verwendet werden, bis der ausgefallene Knoten wiederhergestellt werden kann.

Jeder Amazon DocumentDB-Cluster kann bis zu 15 Replikate unterstützen.

> **Note**
>
> Für maximale Fehlertoleranz sollten Sie Replikate in separaten Availability Zones bereitstellen. Dadurch wird sichergestellt, dass Ihr Amazon DocumentDB-Cluster auch dann weiter funktioniert, wenn eine gesamte Availability Zone nicht mehr verfügbar ist.

Das folgende AWS CLI Beispiel zeigt, wie ein neues Replikat hinzugefügt wird. Der `--availability-zone`-Parameter platziert das Replikat in der angegebenen Availability Zone.

```
aws docdb create-db-instance \
    --db-instance-identifier sample-instance \
    --db-cluster-identifier sample-cluster \
    --engine docdb \
    --db-instance-class db.r5.large \
    --availability-zone us-east-1a
```

Beschreibung von Clustern und Instanzen

Das folgende AWS CLI Beispiel listet alle Amazon DocumentDB-Cluster in einer Region auf. Für bestimmte Verwaltungsfunktionen wie das Cluster- und Instance-Lebenszyklusmanagement nutzt Amazon DocumentDB Betriebstechnologie, die gemeinsam mit Amazon RDS genutzt wird. Der `filterName=engine,Values=docdb` Filterparameter gibt nur Amazon DocumentDB-Cluster zurück.

Weitere Informationen zum Beschreiben und Ändern von Clustern finden Sie im Lebenszyklus des Amazon DocumentDB-Clusters.

```
aws docdb describe-db-clusters --filter Name=engine,Values=docdb
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
    "DBClusters": [
        {
            "AvailabilityZones": [
                "us-east-1c",
                "us-east-1b",
                "us-east-1a"
            ],
            "BackupRetentionPeriod": 1,
            "DBClusterIdentifier": "sample-cluster-1",
            "DBClusterParameterGroup": "sample-parameter-group",
            "DBSubnetGroup": "default",
            "Status": "available",
            ...
        },
        {
            "AvailabilityZones": [
                "us-east-1c",
                "us-east-1b",
                "us-east-1a"
            ],
            "BackupRetentionPeriod": 1,
            "DBClusterIdentifier": "sample-cluster-2",
            "DBClusterParameterGroup": "sample-parameter-group",
            "DBSubnetGroup": "default",
            "Status": "available",
            ...
```

```
        },
        {
            "AvailabilityZones": [
                "us-east-1c",
                "us-east-1b",
                "us-east-1a"
            ],
            "BackupRetentionPeriod": 1,
            "DBClusterIdentifier": "sample-cluster-3",
            "DBClusterParameterGroup": "sample-parameter-group",
            "DBSubnetGroup": "default",
            "Status": "available",
            ...
        }
    ]
}
```

Das folgende AWS CLI Beispiel listet die Instances in einem Amazon DocumentDB-Cluster auf. Weitere Informationen zum Beschreiben und Ändern von Clustern finden Sie im Lebenszyklus einer Amazon DocumentDB DocumentDB-Instanz.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterMembers]'
```

Die Ausgabe sieht wie unten aus. Diese Ausgabe enthält zwei Instances. Die Primär-Instance ist `sample-instance-1` (`"IsClusterWriter": true`). Es gibt auch eine Replikat-Instance, nämlich `sample-instance2` (`"IsClusterWriter: false"`).

```
[
    [
        [
            {
                "DBInstanceIdentifier": "sample-instance-1",
                "IsClusterWriter": true,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            },
            {
                "DBInstanceIdentifier": "sample-cluster-2",
                "IsClusterWriter": false,
                "DBClusterParameterGroupStatus": "in-sync",
```

```
                "PromotionTier": 1
            }
        ]
    ]
]
```

Einen Cluster-Snapshot erstellen

Ein Cluster-Snapshot ist eine vollständige Sicherung der Daten in Ihrem Amazon DocumentDB-Cluster. Wenn der Snapshot erstellt wird, liest Amazon DocumentDB Ihre Daten direkt aus dem Cluster-Volume. Aus diesem Grund können Sie einen Snapshot erstellen, auch wenn zum jeweiligen Zeitpunkt keine Instances in Ihrem Cluster ausgeführt werden. Die zum Erstellen eines Snapshots erforderliche Zeit hängt von der Größe Ihres Cluster-Volumes ab.

Amazon DocumentDB unterstützt automatische Backups, die täglich während des bevorzugten Backup-Fensters — einem Zeitraum von 30 Minuten am Tag — durchgeführt werden. Das folgende AWS CLI Beispiel zeigt, wie Sie das Backup-Fenster für Ihren Cluster anzeigen:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].PreferredBackupWindow'
```

Die Ausgabe zeigt das Sicherungszeitfenster (in UTC):

```
[
    "00:18-00:48"
]
```

Sie können das Backup-Fenster definieren, wenn Sie Ihren Amazon DocumentDB-Cluster erstellen. Sie können das Sicherungszeitfenster auch ändern, wie im folgenden Beispiel gezeigt wird. Wenn Sie kein Backup-Fenster definieren, weist Amazon DocumentDB Ihrem Cluster automatisch eines zu.

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --preferred-backup-window "02:00-02:30"
```

Zusätzlich zu automatischen Sicherungen können Sie jederzeit manuell einen Cluster-Snapshot erstellen. Wenn Sie dies tun, müssen Sie das zu sichernde Cluster festlegen und dem Snapshot

einen eindeutigen Namen geben, damit er später für Wiederherstellungszwecke verwendet werden kann.

Das folgende AWS CLI Beispiel zeigt, wie Sie einen Snapshot Ihrer Daten erstellen.

```
aws docdb create-db-cluster-snapshot \
    --db-cluster-identifier sample-cluster \
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

Wiederherstellung aus einem Snapshot

Sie können einen Cluster-Snapshot in einem neuen Amazon DocumentDB-Cluster wiederherstellen. Dazu geben Sie den Namen des Snapshots und den Namen eines neuen Clusters an. Sie können keine Wiederherstellung aus einem Snapshot in einem vorhandenen Cluster durchführen. Stattdessen erstellt Amazon DocumentDB bei der Wiederherstellung einen neuen Cluster und füllt ihn dann mit Ihren Snapshot-Daten auf.

Das folgende Beispiel zeigt alle Snapshots für den Cluster `sample-cluster`.

```
aws docdb describe-db-cluster-snapshots \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusterSnapshots[*].[DBClusterSnapshotIdentifier,SnapshotType,Status]'
```

Die Ausgabe sieht ungefähr wie folgt aus. Ein manueller Snapshot ist ein Snapshot, den Sie manuell erstellt haben, wohingegen ein automatisierter Snapshot von Amazon DocumentDB innerhalb des Cluster-Backup-Fensters erstellt wird.

```
[
        "sample-cluster-snapshot",
        "manual",
        "available"
    ],
    [
        "rds:sample-cluster",
        "automated",
        "available"
    ]
]
```

Das folgende Beispiel zeigt, wie ein Amazon DocumentDB-Cluster aus einem Snapshot wiederhergestellt wird.

```
aws docdb restore-db-cluster-from-snapshot \
    --engine docdb \
    --db-cluster-identifier new-sample-cluster \
    --snapshot-identifier sample-cluster-snapshot
```

Dem neuen Cluster sind keine Instances zugeordnet. Wenn Sie mit dem Cluster interagieren möchten, müssen Sie ihm also eine Instance hinzufügen.

```
aws docdb create-db-instance \
    --db-instance-identifier new-sample-instance \
    --db-instance-class db.r5.large \
    --engine docdb \
    --db-cluster-identifier new-sample-cluster
```

Sie können die folgenden AWS CLI Operationen verwenden, um den Fortschritt der Cluster- und Instance-Erstellung zu überwachen. Wenn die Cluster- und Instance-Status verfügbar sind, können Sie eine Verbindung zum neuen Endpunkt des Clusters herstellen und auf Ihre Daten zugreifen.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier new-sample-cluster  \
    --query 'DBClusters[*].[Status,Endpoint]'
```

```
aws docdb describe-db-instances \
    --db-instance-identifier new-sample-instance \
    --query 'DBInstances[*].[DBInstanceStatus]'
```

Eine Instanz aus einem Cluster entfernen

Amazon DocumentDB speichert alle Ihre Daten im Cluster-Volume. Die Daten bleiben in diesem Cluster-Volume erhalten. Dies gilt auch dann, wenn Sie alle Instances von Ihrem Cluster entfernen. Wenn Sie erneut auf die Daten zugreifen müssen, können Sie jederzeit dem Cluster eine Instance hinzufügen und da weitermachen, wo Sie aufgehört haben.

Das folgende Beispiel zeigt, wie Sie eine Instance aus Ihrem Amazon DocumentDB-Cluster entfernen.

```
aws docdb delete-db-instance \
    --db-instance-identifier sample-instance
```

Löschen eines Clusters

Bevor Sie einen Amazon DocumentDB-Cluster löschen können, müssen Sie zunächst alle seine Instances entfernen. Das folgende AWS CLI Beispiel gibt Informationen über die Instances in einem Cluster zurück. Wenn diese Operation Instance-Kennungen zurückgibt, müssen Sie die einzelnen Instances löschen. Weitere Informationen finden Sie unter Eine Instanz aus einem Cluster entfernen.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].DBClusterMembers[*].DBInstanceIdentifier'
```

Wenn keine Instances mehr vorhanden sind, können Sie das Cluster löschen. Sie müssen dann eine der folgenden Optionen wählen:

- Einen letzten Snapshot erstellen — Erfassen Sie alle Cluster-Daten in einem Snapshot, sodass Sie später eine neue Instanz mit diesen Daten neu erstellen können. Das Verfahren wird im folgenden Beispiel beschrieben:

  ```
  aws docdb delete-db-cluster \
      --db-cluster-identifier sample-cluster \
      --final-db-snapshot-identifier sample-cluster-snapshot
  ```

- Den letzten Snapshot überspringen — Alle Cluster-Daten werden dauerhaft gelöscht. Diese Aktion ist unwiderruflich. Das Verfahren wird im folgenden Beispiel beschrieben:

  ```
  aws docdb delete-db-cluster \
      --db-cluster-identifier sample-cluster \
      --skip-final-snapshot
  ```

Überblick über die globalen Amazon DocumentDB-Cluster

Was ist ein globaler Cluster?

Ein globaler Cluster besteht aus einer primären Region und bis zu 10 schreibgeschützten sekundären Regionen. Sie geben Schreibvorgänge direkt an den primären Cluster in der primären Region aus und Amazon DocumentDB repliziert die Daten automatisch mithilfe einer dedizierten Infrastruktur in die sekundären Regionen. Die Latenz liegt in der Regel unter einer Sekunde.

Wie sind globale Cluster nützlich?

- Wiederherstellung nach Region-wide Ausfällen — Im Falle eines Region-wide Ausfalls können Sie einen der sekundären Cluster innerhalb von Minuten zu einem primären Cluster hochstufen, wobei ein typisches Recovery Time Objective (RTO) von unter einer Minute liegt. Das Recovery Point Objective (RPO) wird in der Regel in Sekunden gemessen, dies hängt jedoch von der Verzögerung im Netzwerk zum Zeitpunkt des Ausfalls ab.
- Globale Lesevorgänge mit lokaler Latenz — Wenn Sie Niederlassungen auf der ganzen Welt haben, können Sie einen globalen Cluster verwenden, um Ihre wichtigsten Informationsquellen in der Hauptregion auf dem neuesten Stand zu halten. Niederlassungen in Ihren anderen Regionen können mit lokaler Latenz auf die Daten in ihrer eigenen Region zugreifen.
- Skalierbare sekundäre Cluster — Sie können Ihre sekundären Cluster skalieren, indem Sie einer sekundären Region weitere schreibgeschützte Instances hinzufügen. Der sekundäre Cluster ist schreibgeschützt, sodass er bis zu 16 schreibgeschützte Replikatinstanzen unterstützen kann, anstatt wie üblich für einen einzelnen Cluster 15 zu verwenden.
- Schnelle Replikation vom primären zum sekundären Cluster — Die von einem globalen Cluster durchgeführte Replikation hat nur geringe Auswirkungen auf die Leistung des primären Datenbank-Clusters. Die Ressourcen der DB-Instances werden ausschließlich für Lese- und Schreib-Workloads von Anwendungen genutzt.
- Change-Streams — Sie können Change-Streams aus einem sekundären Cluster in globalen Clustern lesen, wodurch die Belastung des primären Clusters reduziert wird und Sie Lese-Workloads aus sekundären Regionen skalieren können.

Was sind die aktuellen Einschränkungen globaler Cluster?

- Globale Cluster werden in Amazon DocumentDB v3.6 nicht unterstützt.
- Globale Cluster werden auf allen Instance-Typen außer db.t3, db.t4g und db.r4 unterstützt.
- Globale Cluster sind in den folgenden Regionen nicht verfügbar: Südamerika (São Paulo), Europa (Mailand), China (Peking) und China (Ningxia).
- Switchover und globales Failover werden nicht unterstützt, wenn Regionen unterschiedliche Engine-Versionen verwenden. Manuelles Failover wird unterstützt, wenn eine Engine-Version nicht übereinstimmt.
- Nur der primäre Cluster führt Schreibvorgänge aus. Clients, die Schreibvorgänge ausführen, stellen eine Verbindung zum Cluster-Endpunkt des primären Clusters her.

- Sie können maximal 10 sekundäre Regionen und eine primäre Region für Ihren Cluster haben.
- Ein sekundärer Cluster kann nicht gestoppt werden. Ein primärer Cluster kann nicht gestoppt werden, wenn ihm sekundäre Cluster zugeordnet sind. Nur ein regionaler Cluster, der keine sekundären Cluster hat, kann gestoppt werden.
- An den sekundären Cluster angehängte Replikate können unter bestimmten Umständen neu gestartet werden. Wenn die Instance der primären Region neu gestartet wird oder ein Failover durchgeführt wird, werden auch die Replikate in der sekundären Region neu gestartet. Der Cluster ist dann nicht verfügbar, bis alle Replikate wieder mit der Writer-Instance des primären Datenbank-Clusters synchronisiert sind. Dieses Verhalten wird erwartet. Vergewissern Sie sich, dass Sie die Auswirkungen auf Ihren globalen Cluster verstanden haben, bevor Sie Änderungen an Ihrem primären Cluster vornehmen.

Themen

- Schnellstartanleitung: globale Cluster
- Verwaltung eines globalen Amazon DocumentDB-Clusters
- Stellen Sie eine Verbindung zu einem globalen Amazon DocumentDB-Cluster her
- Überwachung globaler Amazon DocumentDB-Cluster
- Disaster Recovery und globale Amazon DocumentDB-Cluster

Schnellstartanleitung: globale Cluster

Themen

- Konfiguration
- Einen globalen Amazon DocumentDB-Cluster erstellen
- Hinzufügen eines AWS-Region zu einem globalen Amazon DocumentDB-Cluster
- Verwenden Sie einen Snapshot für Ihren globalen Amazon DocumentDB-Cluster

Konfiguration

Der globale Amazon DocumentDB-Cluster umfasst mindestens zwei. AWS-Regionen Die primäre Region unterstützt einen Cluster mit einer primären (Writer-) Instance und bis zu 15 Replica-Instances, während in einer sekundären Region ein schreibgeschützter Cluster ausgeführt wird, der vollständig aus bis zu 16 Replica-Instances besteht. Ein globaler Cluster kann bis zu 10 sekundäre

Regionen haben. In der Tabelle ist die maximale Anzahl an Clustern, Instances und Replikaten aufgeführt, die in einem globalen Cluster zulässig sind.

Description	Primär AWS-Region	Sekundär AWS-Region
Cluster	1	10 (maximal)
Writer-Inst	1	0
Read-only Instanzen (Amazon DocumentDB-Replikate), pro Cluster	15 (Max.)	16 (Total)
Read-only Instances (maximal zulässig, bei der tatsächlichen Anzahl sekundärer Regionen)	15 - s	s = Gesamtzahl der sekundären AWS-Regionen

Die Cluster haben die folgenden spezifischen Anforderungen:

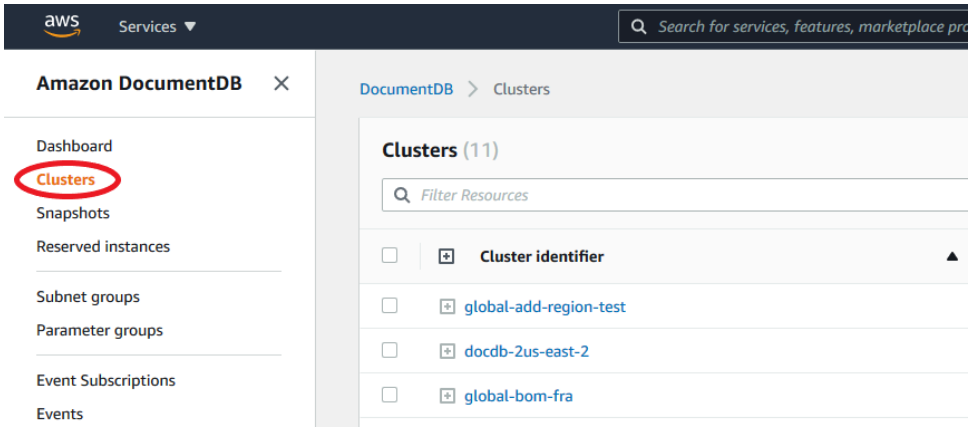
- Anforderungen an die Datenbank-Instance-Klasse — Sie können nur die `db.r6g` Instanzklassen `db.r5` und verwenden.
- AWS-Region Anforderungen — Der primäre Cluster muss sich in einer Region befinden, und mindestens ein sekundärer Cluster muss sich in einer anderen Region desselben Kontos befinden. Sie können bis zu 10 sekundäre (schreibgeschützte) Cluster erstellen, und jeder muss sich in einer anderen Region befinden. Mit anderen Worten, keine zwei Cluster können sich in derselben Region befinden.
- Anforderungen an die Benennung — Die Namen, die Sie für jeden Ihrer Cluster wählen, müssen in allen Regionen eindeutig sein. Sie können nicht denselben Namen für verschiedene Cluster verwenden, obwohl sie sich in verschiedenen Regionen befinden.

Einen globalen Amazon DocumentDB-Cluster erstellen

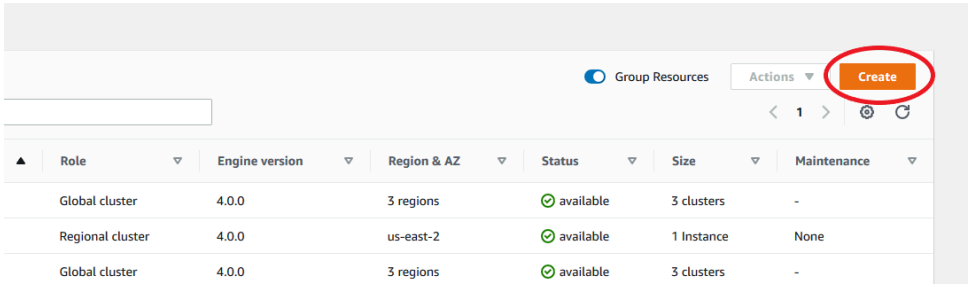
Sind Sie bereit, Ihren ersten globalen Cluster aufzubauen? In diesem Abschnitt erklären wir, wie Sie einen brandneuen globalen Cluster mit neuen Datenbankclustern und Instances erstellen, indem Sie entweder die AWS-Managementkonsole oder AWS CLI die folgenden Anweisungen verwenden.

Verwendung der AWS-Managementkonsole

1. Navigieren Sie im AWS-Managementkonsole zu Amazon DocumentDB.
2. Wenn Sie zur Amazon DocumentDB-Konsole gelangen, wählen Sie Clusters.

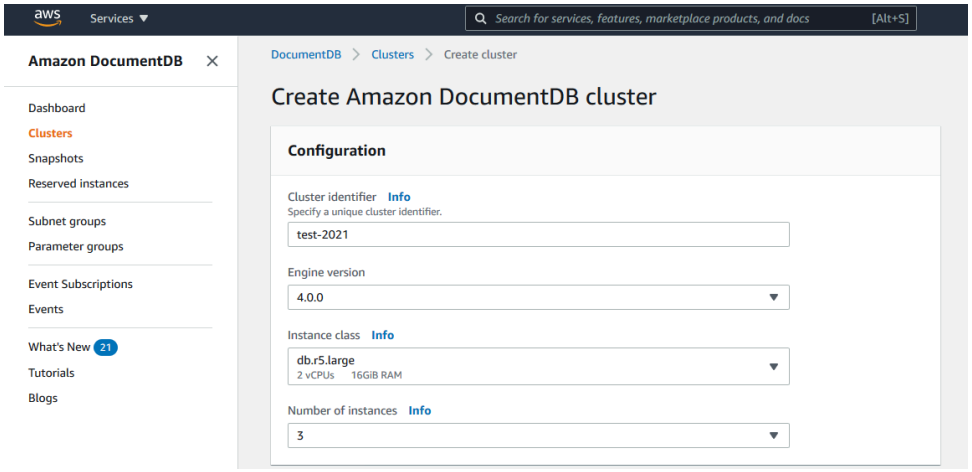


3. Wählen Sie Erstellen aus.

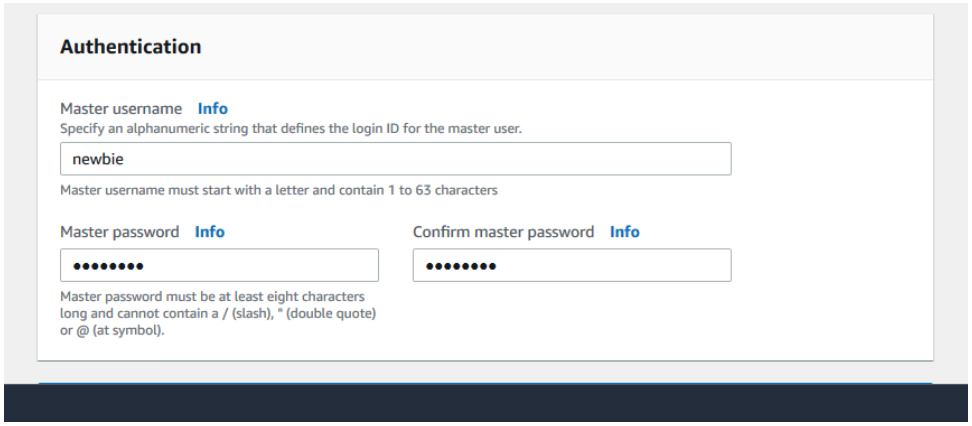


4. Füllen Sie den Abschnitt „Konfiguration“ des Formulars „Amazon DocumentDB-Cluster erstellen“ entsprechend aus:

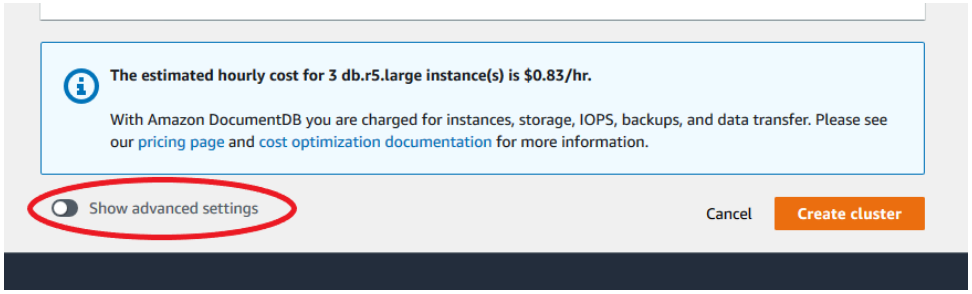
   - Cluster-ID: Sie können entweder eine eindeutige ID für diese Instance eingeben oder Amazon DocumentDB erlauben, die Instance-ID auf der Grundlage der Cluster-ID bereitzustellen.
   - Engine-Version: Wählen Sie 4.0.0
   - Instanzklasse: Wählen Sie db.r5.large
   - Anzahl der Instanzen: Wählen Sie 3.



5. Geben Sie im Abschnitt Authentifizierung einen Master-Benutzernamen und ein Master-Passwort ein.

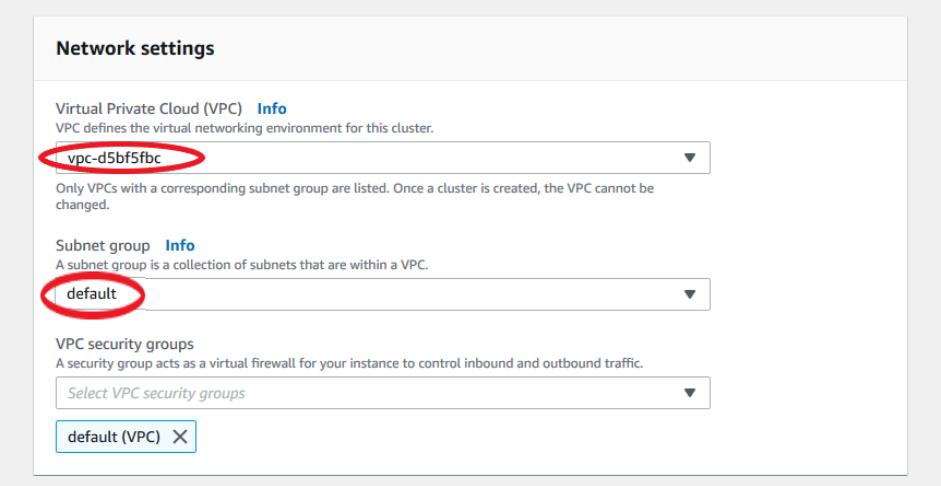


6. Wählen Sie Erweiterte Einstellungen anzeigen.

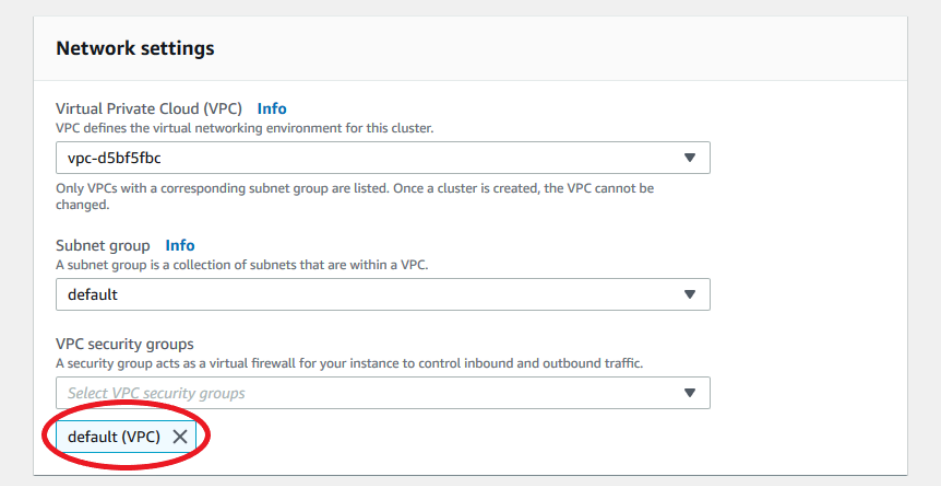


7. Im Abschnitt Netzwerkeinstellungen:

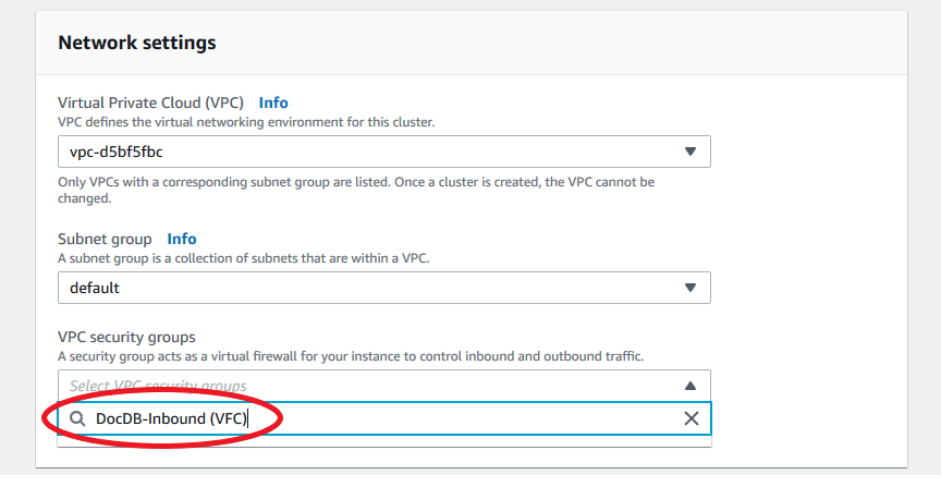
   - Behalten Sie die Standardoptionen für Virtual Private Cloud (VPC) und Subnetzgruppe bei.



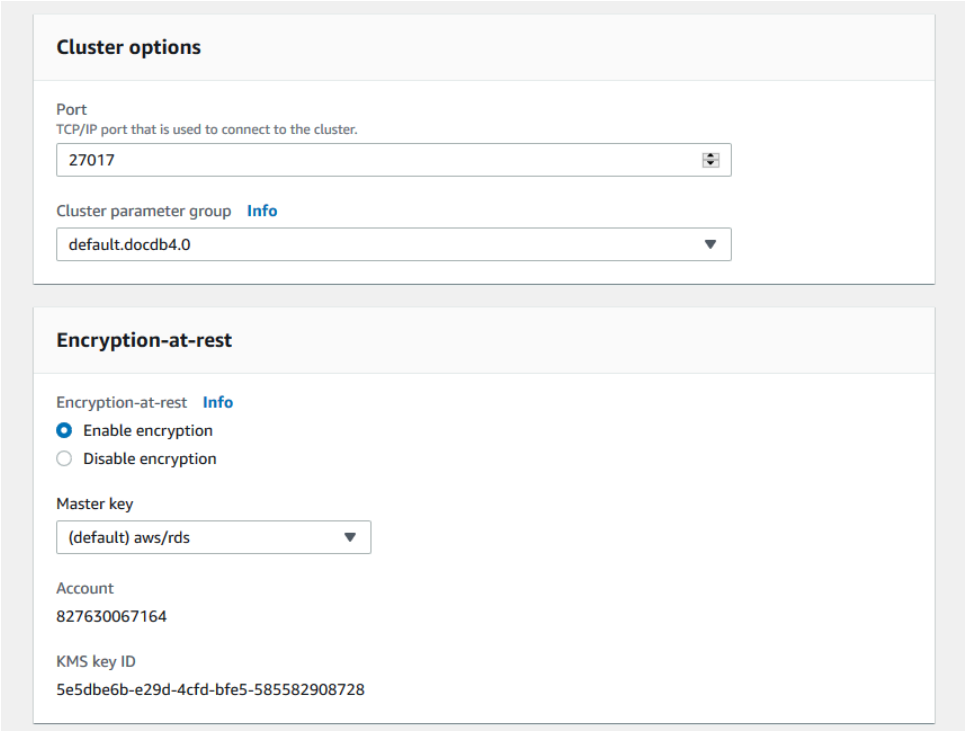
- Für VPC-Sicherheitsgruppen sollte die Standardeinstellung (VPC) bereits hinzugefügt werden.



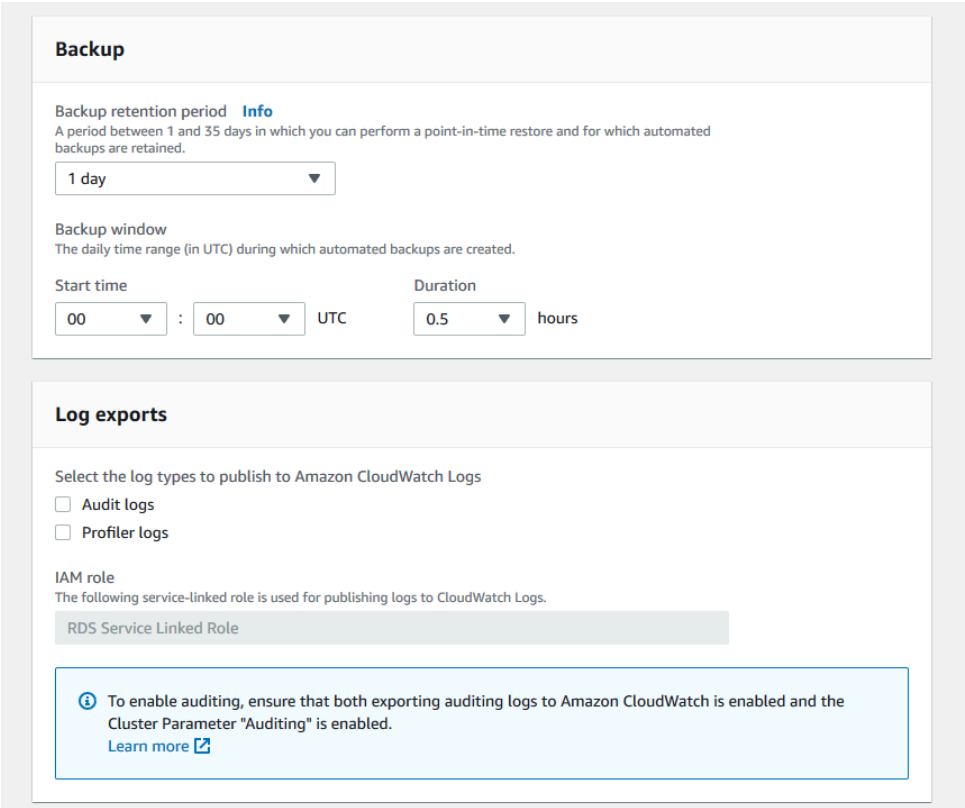
- Geben Sie den DocDB Text in das Feld VPC-Sicherheitsgruppen ein und wählen Sie DocDB-Inbound (VFC).



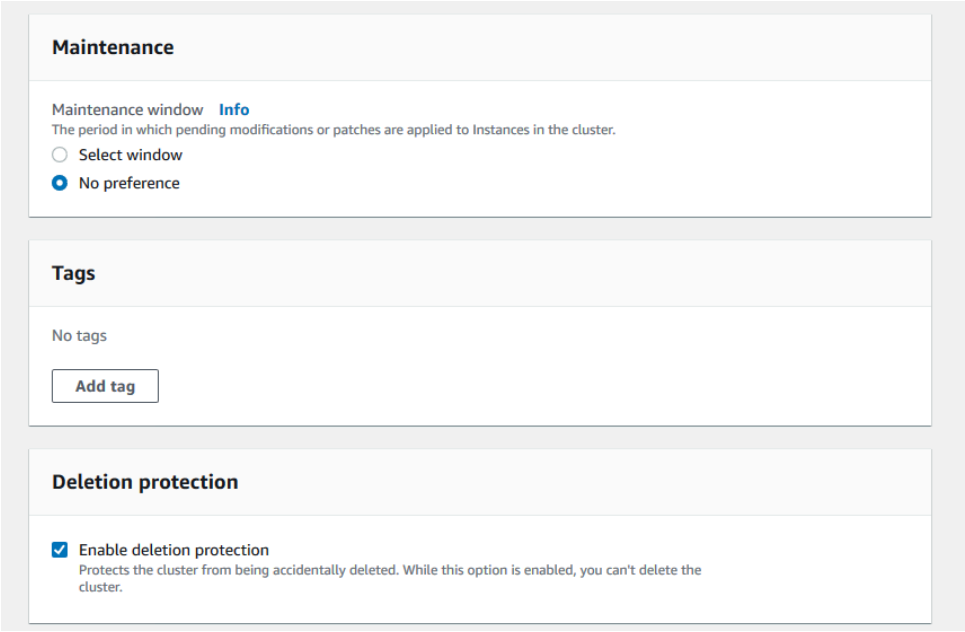
8. Belassen Sie für Cluster-Optionen und Encryption-at-rest die Standardauswahl.



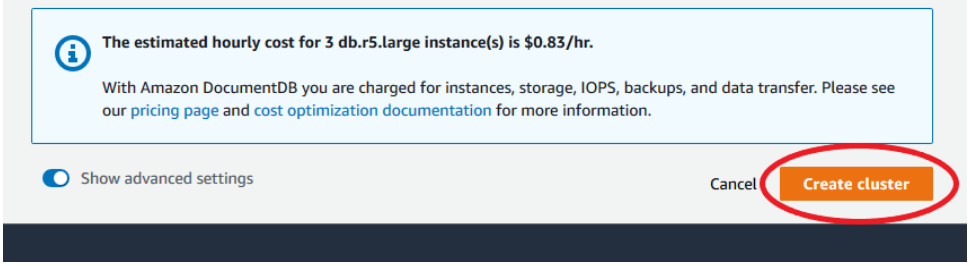
9. Belassen Sie bei Backup - und Log-Exporten die Standardauswahl.



10. Belassen Sie die Standardoptionen für Wartung , Stichwörter und Löschschutz.



11. Wählen Sie nun Cluster erstellen.



Verwendung der AWS CLI

Um einen Amazon DocumentDB Regional Cluster zu erstellen, rufen Sie AWS CLI create-global-cluster auf. Mit dem folgenden AWS CLI Befehl wird ein Amazon DocumentDB-Cluster mit dem Namen erstellt. `global-cluster-id` Weitere Informationen zum Löschschutz finden Sie unterLöschen eines Amazon DocumentDB-Clusters.

Außerdem `--engine-version` ist dies ein optionaler Parameter, der standardmäßig auf die neueste Version der Hauptengine zurückgreift. Die aktuelle Standard-Engine-Version ist `5.0.0` (Hinweis: Amazon DocumentDB 8.0 ist verfügbar, muss aber explizit als `8.0.0` angegeben werden). Wenn neue Hauptversionen der Engine veröffentlicht werden, `--engine-version` wird die Standard-Engine-Version für aktualisiert, sodass sie die letzte Hauptversion der Engine wiedergibt. Geben Sie daher für Produktions-Workloads und insbesondere für solche, die von Scripting, Automatisierung oder CloudFormation Vorlagen abhängen, explizit die gewünschte Hauptversion `--engine-version` an.

Wenn ein `db-subnet-group-name` oder nicht angegeben `vpc-security-group-id` ist, verwendet Amazon DocumentDB die Standard-Subnetzgruppe und die Amazon VPC-Sicherheitsgruppe für die angegebene Region.

Ersetzen Sie im folgenden Beispiel jede durch Ihre *`user input placeholder`* eigenen Informationen.

Für Linux, macOS oder Unix:

```
aws docdb create-db-cluster \
      --global-cluster-identifier global-cluster-id \
      --source-db-cluster-identifier arn:aws:rds:us-east-1:111122223333:cluster-id
```

Für Windows:

```
aws docdb create-db-cluster ^
      --global-cluster-identifier global-cluster-id ^
      --source-db-cluster-identifier arn:aws:rds:us-east-1:111122223333:cluster-id
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    "DBCluster": {
        "StorageEncrypted": false,
        "DBClusterMembers": [],
        "Engine": "docdb",
        "DeletionProtection" : "enabled",
        "ClusterCreateTime": "2018-11-26T17:15:19.885Z",
        "DBSubnetGroup": "default",
        "EngineVersion": "4.0.0",
        "MasterUsername": "masteruser",
        "BackupRetentionPeriod": 1,
        "DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:cluster-id",
        "DBClusterIdentifier": "cluster-id",
        "MultiAZ": false,
        "DBClusterParameterGroup": "default.docdb4.0",
        "PreferredBackupWindow": "09:12-09:42",
        "DbClusterResourceId": "cluster-KQSGI4MHU4NTDDRVNLNTU7XVAY",
        "PreferredMaintenanceWindow": "tue:04:17-tue:04:47",
        "Port": 27017,
```

```
        "Status": "creating",
        "ReaderEndpoint": "cluster-id.cluster-ro-sfcrlcjcoroz.us-
east-1.docdb.amazonaws.com",
        "AssociatedRoles": [],
        "HostedZoneId": "ZNKXTT8WH85VW",
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-77186e0d",
                "Status": "active"
            }
        ],
        "AvailabilityZones": [
            "us-east-1a",
            "us-east-1c",
            "us-east-1e"
        ],
        "Endpoint": "cluster-id.cluster-sfcrlcjcoroz.us-east-1.docdb.amazonaws.com"
    }
}
```

Die Erstellung des Clusters dauert mehrere Minuten. Sie können das AWS-Managementkonsole oder verwenden AWS CLI , um den Status Ihres Clusters zu überwachen. Weitere Informationen finden Sie unter Überwachung des Status eines Amazon DocumentDB-Clusters.

> **Important**
>
> Wenn Sie das verwenden AWS CLI , um einen Amazon DocumentDB Regional Cluster zu erstellen, werden keine Instances erstellt. Daher müssen Sie explizit eine primäre Instance und alle benötigten Replikat-Instances anlegen. Sie können entweder die Konsole verwenden oder AWS CLI um die Instances zu erstellen. Weitere Informationen finden Sie unter Hinzufügen einer Amazon DocumentDB DocumentDB-Instance zu einem Cluster und CreateDBCluster in der Amazon DocumentDB-API-Referenz.

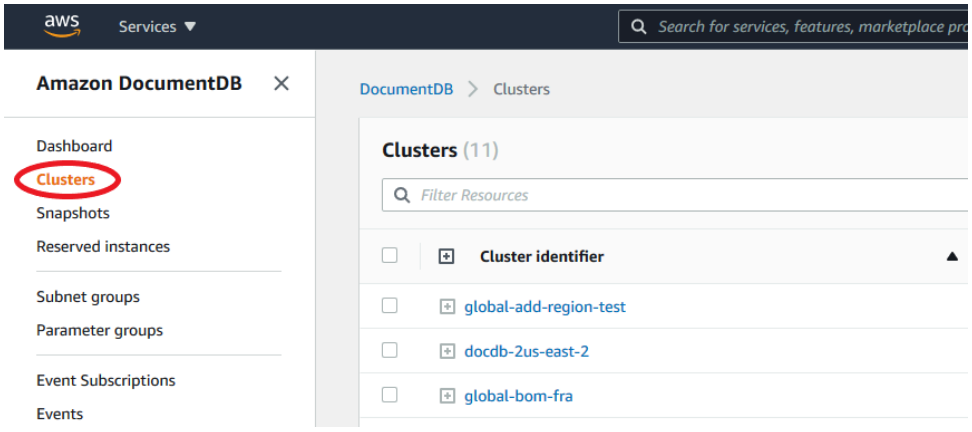
Sobald Ihr regionaler Cluster verfügbar ist, können Sie mithilfe der folgenden Anweisungen einen sekundären Cluster in einer anderen Region hinzufügen:. Hinzufügen eines AWS-Region zu einem globalen Amazon DocumentDB-Cluster Wenn Sie eine Region hinzufügen, wird Ihr regionaler Cluster zu Ihrem primären Cluster, und Sie haben einen neuen sekundären Cluster in der von Ihnen ausgewählten Region.

Hinzufügen eines AWS-Region zu einem globalen Amazon DocumentDB-Cluster

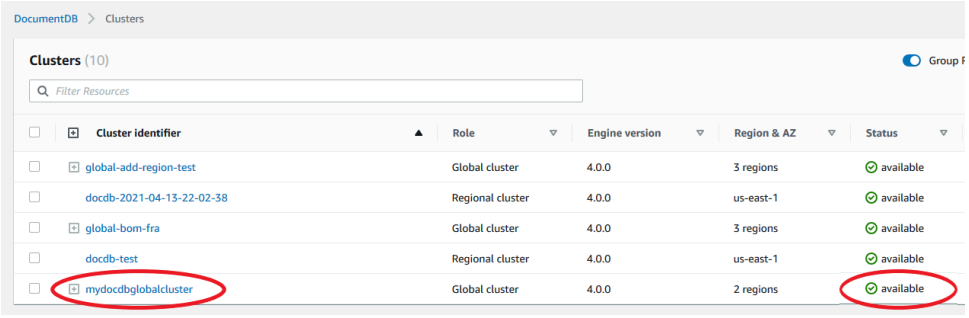
Ein globaler Cluster benötigt mindestens einen sekundären Cluster in einer anderen Region als der primäre Cluster, und Sie können bis zu 10 sekundäre Cluster hinzufügen. Beachten Sie, dass Sie für jeden sekundären Cluster, den Sie hinzufügen, die Anzahl der im primären Cluster zulässigen Replikate um eins reduzieren müssen. Wenn Ihr globaler Cluster beispielsweise 10 sekundäre Regionen hat, kann Ihr primärer Cluster nur 5 (statt 15) Replikate haben. Weitere Informationen finden Sie unter Konfiguration.

Verwendung der AWS-Managementkonsole

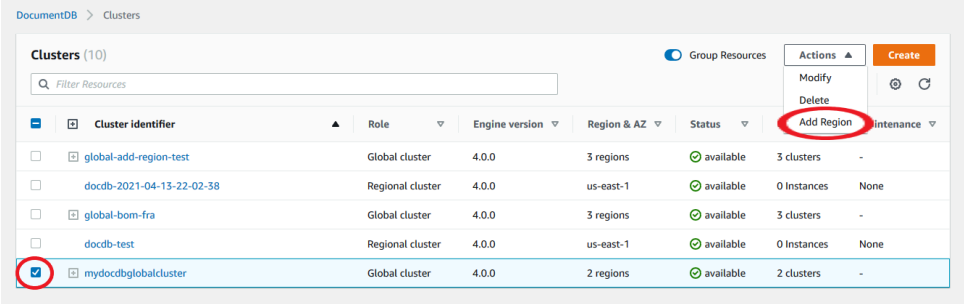
1. Melden Sie sich bei der Amazon DocumentDB-Konsole an AWS-Managementkonsole und öffnen Sie sie.
2. Klicken Sie im Navigationsbereich auf Cluster.



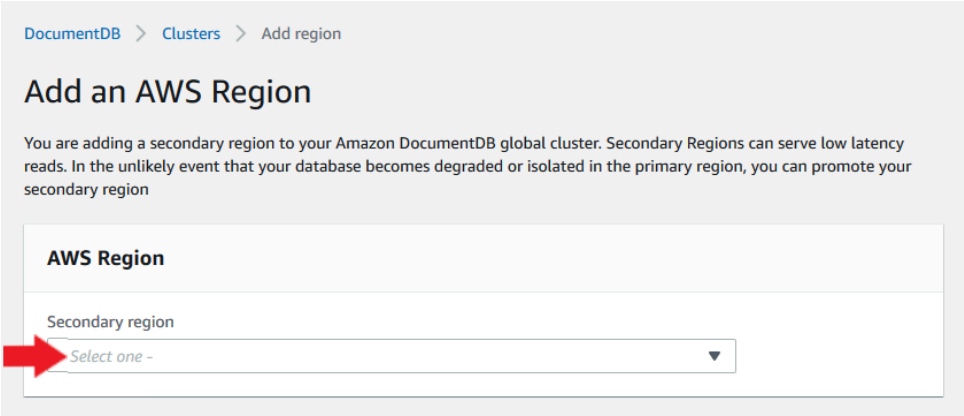
3. Wählen Sie den Cluster aus, zu dem Sie einen sekundären Cluster hinzufügen möchten. Stellen Sie sicher, dass der `Available` Cluster



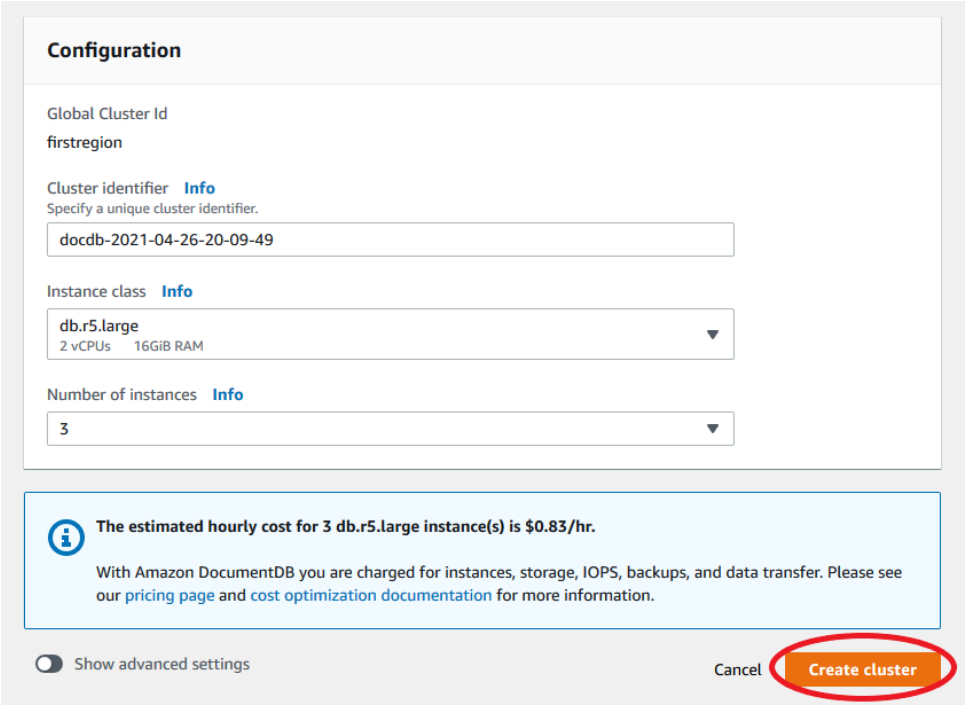
4. Wählen Sie die Dropdownliste für Aktionen aus und wählen Sie dann Region hinzufügen aus.



5. Wählen Sie auf der AWS-Region Seite „Eine hinzufügen“ die sekundäre Region aus. Beachten Sie, dass Sie keine Region auswählen können, die bereits über einen sekundären Cluster für denselben globalen Cluster verfügt. Außerdem kann es sich nicht um dieselbe Region wie der primäre Cluster handeln. Wenn dies die erste Region ist, die Sie hinzufügen, müssen Sie auch eine globale Cluster-ID Ihrer Wahl angeben.



6. Füllen Sie die verbleibenden Felder für den sekundären Cluster in der neuen Region aus und wählen Sie dann Cluster erstellen aus. Nachdem Sie die Region hinzugefügt haben, können Sie sie in der Liste der Cluster im sehen AWS-Managementkonsole.



Verwendung der AWS CLI

- Verwenden Sie den `create-db-cluster` CLI-Befehl mit dem Namen (`--global-cluster-identifier`) Ihres globalen Clusters. Für andere Parameter, führen Sie die folgenden Schritte aus:
  - Wählen Sie für `--region` eine andere Region AWS-Region als die Ihrer Hauptregion.
  - Wählen Sie bestimmte Werte für die Parameter `--engine-version` und `--engine` aus.
  - Für einen verschlüsselten Cluster geben Sie Ihr primäres Cluster AWS-Region als das `--source-region` für die Verschlüsselung an.

Das folgende Beispiel erstellt einen neuen Amazon DocumentDB-Cluster und hängt ihn als schreibgeschützten sekundären Cluster an den globalen Cluster an. Im letzten Schritt wird die Instance dem neuen Cluster hinzugefügt.

Ersetzen Sie im folgenden Beispiel jede *`user input placeholder`* durch Ihre eigenen Informationen.

Für Linux, macOS oder Unix:

```
aws docdb --region secondary-region-id \
```

```
  create-db-cluster \
    --db-cluster-identifier cluster-id \
    --global-cluster-identifier global-cluster-id \
    --engine-version version \
    --engine docdb

aws docdb --region secondary-region-id \
  create-db-instance \
    --db-cluster-identifier cluster-id \
    --global-cluster-identifier global-cluster-id \
    --engine-version version \
    --engine docdb
```

Für Windows:

```
aws docdb --region secondary-region-id ^
  create-db-cluster ^
    --db-cluster-identifier cluster-id ^
    --global-cluster-identifier global-cluster-id ^
    --engine-version version ^
    --engine docdb

aws docdb --region secondary-region-id ^
  create-db-instance ^
    --db-cluster-identifier cluster-id ^
    --global-cluster-identifier global-cluster-id ^
    --engine-version version ^
    --engine docdb
```

Verwenden Sie einen Snapshot für Ihren globalen Amazon DocumentDB-Cluster

Sie können einen Snapshot eines Amazon DocumentDB-Clusters wiederherstellen, um ihn als Ausgangspunkt für Ihren globalen Cluster zu verwenden. Dazu müssen Sie den Snapshot wiederherstellen und einen neuen Cluster erstellen. Dieser dient als primärer Cluster Ihres globalen Clusters. Anschließend können Sie dem wiederhergestellten Cluster eine weitere Region hinzufügen und ihn so in einen globalen Cluster umwandeln.

Verwaltung eines globalen Amazon DocumentDB-Clusters

Sie führen die meisten Verwaltungsvorgänge auf den einzelnen Clustern durch, aus denen sich ein globaler Cluster zusammensetzt. Wenn Sie in der Konsole auf der Seite Cluster die Option Verwandte Ressourcen gruppieren auswählen, werden der primäre Cluster und die sekundären Cluster unter dem zugehörigen globalen Cluster gruppiert angezeigt.

Auf der Registerkarte „Konfiguration“ für einen globalen Cluster werden der AWS-Regionen Ort, an dem die Cluster ausgeführt werden, die Version und die globale Cluster-ID angezeigt.

Themen

- Ändern eines globalen Amazon DocumentDB-Clusters
- Ändern von Parametern in einem globalen Amazon DocumentDB-Cluster
- Einen Cluster aus einem globalen Amazon DocumentDB-Cluster entfernen
- Löschen eines Clusters aus einem globalen Amazon DocumentDB-Cluster
- Erstellen eines Headless-Amazon DocumentDB-Clusters in einer sekundären Region

Ändern eines globalen Amazon DocumentDB-Clusters

Auf der Seite „Cluster“ AWS-Managementkonsole werden alle Ihre globalen Cluster aufgeführt, wobei jeweils der primäre Cluster und der sekundäre Cluster angezeigt werden. Der globale Cluster hat seine eigenen Konfigurationseinstellungen. Insbesondere hat er Regionen, die seinen primären und sekundären Clustern zugeordnet sind.

Wenn Sie Änderungen am globalen Cluster vornehmen, haben Sie die Möglichkeit, die Änderungen rückgängig zu machen.

Wenn Sie Weiter wählen, bestätigen Sie die Änderungen.

Ändern von Parametern in einem globalen Amazon DocumentDB-Cluster

Sie können die Cluster-Parametergruppen unabhängig für jeden Cluster innerhalb des globalen Clusters konfigurieren. Die meisten Parameter funktionieren genauso wie für andere Arten von Amazon DocumentDB-Clustern. Sorgen Sie dafür, dass die Einstellungen aller Cluster in einer globalen Datenbank konsistent sind, um unerwartete Verhaltensänderungen zu vermeiden, wenn Sie einen sekundären Cluster zum primären Cluster heraufstufen.

Sie sollten z. B. die gleichen Einstellungen für Zeitzonen und Zeichensätze verwenden, um inkonsistentes Verhalten zu vermeiden, wenn ein anderer Cluster die Rolle des primären Clusters übernimmt.

Einen Cluster aus einem globalen Amazon DocumentDB-Cluster entfernen

Es gibt verschiedene Situationen, in denen Sie Cluster aus Ihrem globalen Cluster entfernen möchten. Beispielsweise möchten Sie möglicherweise einen Cluster aus einem globalen Cluster entfernen, wenn der primäre Cluster heruntergestuft oder isoliert wird. Es wird dann zu einem eigenständigen bereitgestellten Cluster, der zum Erstellen eines neuen globalen Clusters verwendet werden kann. Weitere Informationen hierzu finden Sie unter Durchführung eines manuellen Failovers für einen globalen Amazon DocumentDB-Cluster.

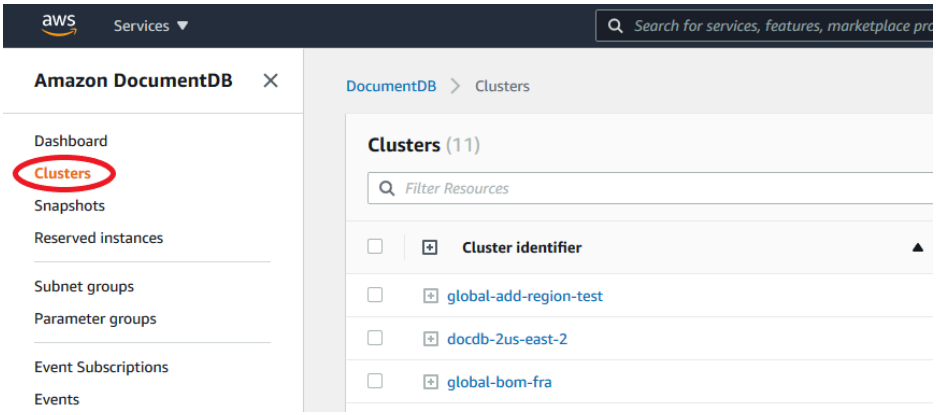
Möglicherweise möchten Sie auch Cluster entfernen, weil Sie einen globalen Cluster löschen möchten, den Sie nicht mehr benötigen. Sie können den globalen Cluster erst löschen, nachdem Sie alle zugehörigen Cluster getrennt haben und der primäre Cluster als letzter übrig bleibt. Weitere Informationen finden Sie unter Löschen eines Clusters aus einem globalen Amazon DocumentDB-Cluster.

> **Note**
>
> Wenn ein Cluster vom globalen Cluster getrennt wird, wird er nicht mehr mit dem primären Cluster synchronisiert. Es wird zu einem eigenständigen bereitgestellten Cluster mit vollem read/write Funktionsumfang. Außerdem ist es in der Amazon DocumentDB-Konsole nicht mehr sichtbar. Es ist nur sichtbar, wenn Sie in der Konsole die Region auswählen, in der sich der Cluster befand.

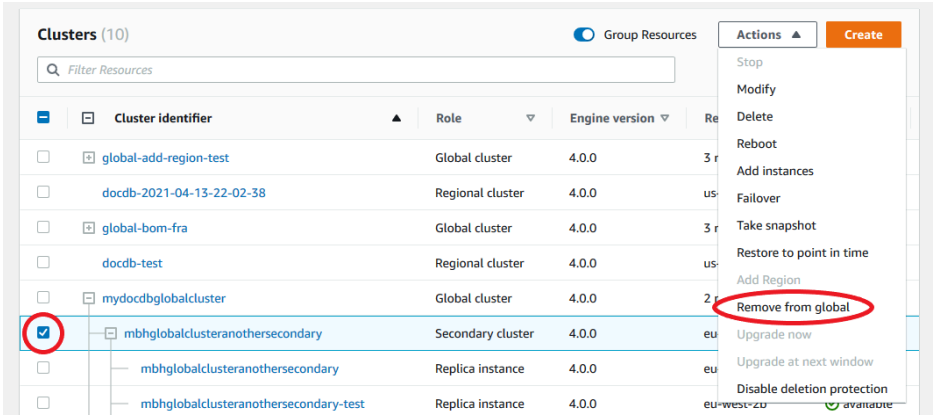
Sie können Cluster mithilfe der AWS-Managementkonsole, der oder der RDS-API aus Ihrem globalen Cluster entfernen. AWS CLI

Using the AWS-Managementkonsole

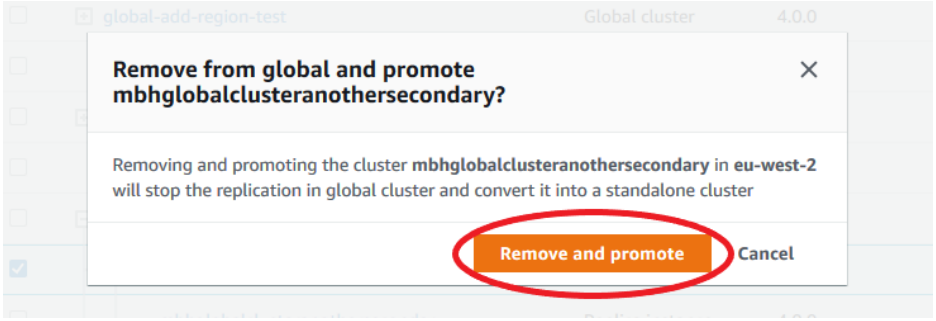
1. Melden Sie sich bei der an AWS-Managementkonsole und navigieren Sie zur Amazon DocumentDB-Konsole.
2. Wählen Sie in der linken Navigationsleiste Clusters aus.



3. Erweitern Sie den globalen Cluster, sodass Sie alle sekundären Cluster sehen können. Wählen Sie die sekundären Cluster aus, die Sie entfernen möchten. Wählen Sie Aktionen und wählen Sie im Dropdownmenü die Option Aus der globalen Liste entfernen aus.



4. Es wird eine Aufforderung angezeigt, in der Sie aufgefordert werden, zu bestätigen, dass Sie den sekundären Cluster vom globalen Cluster trennen möchten. Wählen Sie Entfernen und heraufstufen, um den Cluster aus dem globalen Cluster zu entfernen.



Jetzt dient dieser Cluster nicht mehr als sekundärer Cluster und ist nicht mehr mit dem primären Cluster synchronisiert. Es handelt sich um einen eigenständigen Cluster mit vollem read/write Funktionsumfang.

Nachdem Sie alle sekundären Cluster entfernt oder gelöscht haben, können Sie den primären Cluster auf die gleiche Weise entfernen. Sie können den primären Cluster erst trennen oder vom globalen Cluster entfernen, nachdem Sie alle sekundären Cluster entfernt haben. Der globale Cluster verbleibt möglicherweise in der Cluster-Liste und enthält keine Regionen und AZs. Sie können ihn löschen, wenn Sie diesen globalen Cluster nicht mehr verwenden möchten.

Using the AWS CLI

Um einen Cluster aus einem globalen Cluster zu entfernen, führen Sie den `remove-from-global-cluster` CLI-Befehl mit den folgenden Parametern aus:

- `--global-cluster-identifier`— Der Name (Identifier) Ihres globalen Clusters.
- `--db-cluster-identifier`— Der Name jedes Clusters, das aus dem globalen Cluster entfernt werden soll.

In den folgenden Beispielen wird zuerst ein sekundärer Cluster und dann der primäre Cluster aus einem globalen Cluster entfernt.

Für Linux, macOS oder Unix:

```
aws docdb --region secondary_region \
  remove-from-global-cluster \
    --db-cluster-identifier secondary_cluster_ARN \
    --global-cluster-identifier global_cluster_id

aws docdb --region primary_region \
  remove-from-global-cluster \
    --db-cluster-identifier primary_cluster_ARN \
    --global-cluster-identifier global_cluster_id
```

Wiederholen Sie den `remove-from-global-cluster --db-cluster-identifier secondary_cluster_ARN` Befehl für jede sekundäre Region in Ihrem globalen Cluster.

Für Windows:

```
aws docdb --region secondary_region ^
  remove-from-global-cluster ^
    --db-cluster-identifier secondary_cluster_ARN ^
    --global-cluster-identifier global_cluster_id
```

```
aws docdb --region primary_region ^
  remove-from-global-cluster ^
    --db-cluster-identifier primary_cluster_ARN ^
    --global-cluster-identifier global_cluster_id
```

Wiederholen Sie den `remove-from-global-cluster --db-cluster-identifier secondary_cluster_ARN` Befehl für jede sekundäre Region in Ihrem globalen Cluster.

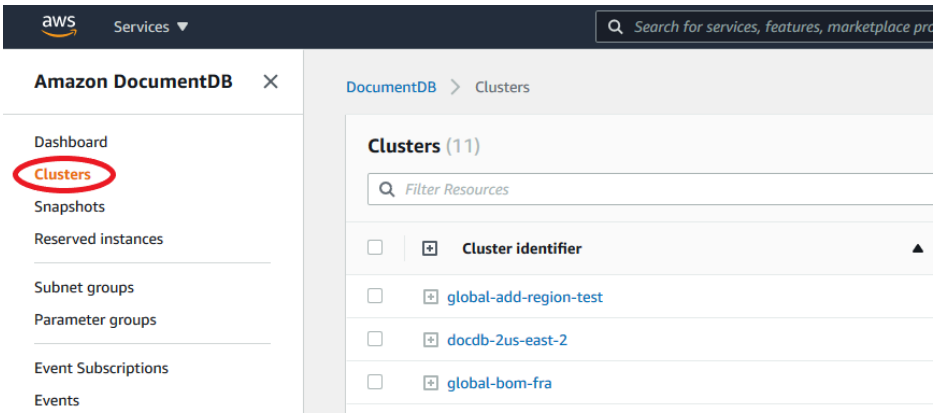
Löschen eines Clusters aus einem globalen Amazon DocumentDB-Cluster

Gehen Sie wie folgt vor, um einen globalen Cluster zu löschen:

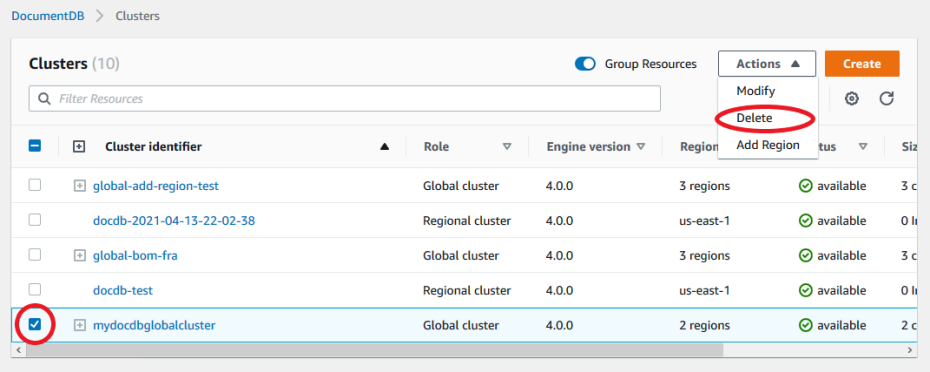
- Entfernen Sie alle sekundären Cluster aus dem globalen Cluster. Jeder Cluster wird zu einem eigenständigen Cluster. Weitere Informationen finden Sie im vorherigen Abschnitt,Einen Cluster aus einem globalen Amazon DocumentDB-Cluster entfernen.
- Löschen Sie aus jedem Standalone-Cluster alle Replikate.
- Entfernen Sie den primären Cluster aus dem globalen Cluster. Dies wird zu einem eigenständigen Cluster.
- Löschen Sie aus dem primären Cluster zuerst alle Replikate und dann die primäre Instance. Durch das Löschen der primären Instance aus dem neuen Standalone-Cluster werden in der Regel auch der Cluster und der globale Cluster entfernt.

Using the AWS-Managementkonsole

1. Melden Sie sich bei der an AWS-Managementkonsole und navigieren Sie zur Amazon DocumentDB-Konsole.
2. Wählen Sie Clusters und suchen Sie den globalen Cluster, den Sie löschen möchten.



3. Wählen Sie bei ausgewähltem globalen Cluster im Menü „Aktionen“ die Option Löschen aus.



Vergewissern Sie sich, dass alle Cluster aus dem globalen Cluster entfernt wurden. Der globale Cluster sollte keine Regionen und AZs und eine Clustergröße von Null aufweisen. Wenn der globale Cluster Cluster enthält, können Sie ihn noch nicht löschen. Sie müssen zuerst die Anweisungen im vorherigen Schritt befolgen Einen Cluster aus einem globalen Amazon DocumentDB-Cluster entfernen.

Using the AWS CLI

Um einen globalen Cluster zu löschen, führen Sie den `delete-global-cluster` CLI-Befehl mit dem Namen AWS-Region und der globalen Cluster-ID aus, wie im folgenden Beispiel gezeigt.

Für Linux, macOS oder Unix:

```
aws docdb --region primary_region delete-global-cluster \
   --global-cluster-identifier global_cluster_id
```

Für Windows:

```
aws docdb --region primary_region delete-global-cluster ^
   --global-cluster-identifier global_cluster_id
```

Erstellen eines Headless-Amazon DocumentDB-Clusters in einer sekundären Region

Obwohl ein globaler Amazon DocumentDB-Cluster mindestens einen sekundären Cluster in einem anderen AWS-Region als dem primären Cluster benötigt, können Sie eine Headless-Konfiguration für den sekundären Cluster verwenden. Ein sekundärer Amazon DocumentDB-Cluster ohne Headless ist ein Cluster ohne Instance. Diese Art der Konfiguration kann die Kosten für einen globalen Cluster

senken. In einem Amazon DocumentDB-Cluster sind Rechenleistung und Speicher entkoppelt. Ohne die Instance wird Ihnen keine Rechenleistung berechnet, sondern nur die Speicherung. Wenn es richtig eingerichtet ist, wird das Speicher-Volume eines sekundären Headless-Clusters mit dem primären Cluster synchronisiert.

Sie fügen den sekundären Cluster wie gewohnt hinzu, wenn Sie einen globalen Amazon DocumentDB-Cluster erstellen. Nachdem der primäre Cluster jedoch mit der Replikation auf den sekundären Cluster begonnen hat, löschen Sie die schreibgeschützte Instance aus dem sekundären Cluster. Dieser sekundäre Cluster gilt jetzt als „headless“, da er keine Instanz mehr hat. Das Speichervolume wird jedoch mit dem primären Amazon DocumentDB-Cluster synchronisiert.

> **Important**
>
> Verwenden Sie Headless-Cluster nur, wenn Sie Region-wide Ausfälle für 15 Minuten oder länger tolerieren können. Für die Wiederherstellung nach einem Region-wide Ausfall mit einem sekundären Headless-Cluster müssen Sie nach einem Failover eine neue Instance erstellen. Es kann 10—15 Minuten dauern, bis eine neue Instanz verfügbar ist.

So fügen Sie Ihrem globalen Cluster einen sekundären Headless-Cluster hinzu

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Amazon DocumentDB-Konsole.
2. Wählen Sie in der linken Navigationsleiste Clusters aus.
3. Wählen Sie den globalen Cluster aus, der einen sekundären Cluster benötigt. Stellen Sie sicher, dass der primäre `Available` Cluster
4. Wählen Sie für Aktionen die Option Region hinzufügen aus.
5. Wählen Sie auf der Seite „Region hinzufügen“ die sekundäre Region aus.

   > **Note**
   >
   > Sie können keine Region auswählen, die bereits über einen sekundären Cluster für denselben globalen Cluster verfügt. Außerdem kann es sich nicht um dieselbe Region wie der primäre Cluster handeln.

6. Füllen Sie die restlichen Felder für den sekundären -Cluster in der neuen -Region aus. Dies sind dieselben Konfigurationsoptionen wie für jede Clusterinstanz.

7. Fügen Sie eine Region hinzu. Nachdem Sie das Hinzufügen der Region zu Ihrem globalen Cluster abgeschlossen haben, erscheint sie `Clusters` in der Liste der AWS-Managementkonsole.
8. Überprüfen Sie den Status des sekundären Clusters und seiner Reader-Instance, bevor Sie fortfahren. Verwenden Sie AWS-Managementkonsole dazu die oder AWS CLI. Hier ist ein Beispielbefehl, wenn Sie den verwenden AWS CLI:

   ```
   $ aws docdb describe-db-clusters --db-cluster-identifier secondary-cluster-id --
   query '*[].[Status]' --output text
   ```

   Es kann mehrere Minuten dauern, bis der Status eines neu hinzugefügten sekundären Clusters von „Erstellt“ zu „Verfügbar“ wechselt. Wenn der Cluster verfügbar ist, können Sie die Reader-Instanz löschen.
9. Wählen Sie die Reader-Instance im sekundären Cluster aus und klicken Sie dann auf Löschen.
10. Nach dem Löschen der Reader-Instance bleibt der sekundäre Cluster Teil des globalen Clusters. Ihm sollte keine Instanz zugeordnet sein.

> **Note**
>
> Sie können diesen kopflosen sekundären Amazon DocumentDB-Cluster verwenden, um Ihren globalen Amazon DocumentDB-Cluster nach einem ungeplanten Ausfall in der primären Region manuell wiederherzustellen, falls ein solcher Ausfall auftritt.

Stellen Sie eine Verbindung zu einem globalen Amazon DocumentDB-Cluster her

Wie Sie eine Verbindung zu einem globalen Cluster herstellen, hängt davon ab, ob Sie in den Cluster schreiben oder aus dem Cluster lesen müssen:

- Für schreibgeschützte Anfragen oder Abfragen stellen Sie eine Verbindung zum Reader-Endpunkt für den Cluster in Ihrem her. AWS-Region
- Stellen Sie zum Ausführen von Data Manipulation Language (DML)- und Data Definition Language (DDL)-Anweisungen eine Verbindung zum Cluster-Endpunkt des primären Clusters her. Dieser Endpunkt befindet sich möglicherweise in einer anderen Anwendung AWS-Region als Ihrer Anwendung.

Wenn Sie einen globalen Cluster in der Konsole anzeigen, können Sie alle Allzweck-Endpunkte sehen, die mit all seinen Clustern verknüpft sind.

Wie Sie eine Verbindung zu einem globalen Cluster herstellen, hängt davon ab, ob Sie in die Datenbank schreiben oder aus der Datenbank lesen müssen. Für DDL-, DML- und Lesevorgänge, die Sie von der primären Region aus ausführen möchten, stellen Sie mithilfe des Cluster-Endpunkts im Replikatsatzmodus eine Verbindung zu Ihrem primären Cluster her, wobei die Lesepräferenz von. `secondaryPreferred=true` Dadurch wird der Schreibdatenverkehr an die Writer-Instance Ihres primären Clusters und der Lese-Traffic an die Replikat-Instance Ihres primären Clusters weitergeleitet.

Stellen Sie für regionsübergreifenden, schreibgeschützten Datenverkehr über den Cluster-Endpunkt im Replikatsatzmodus eine Verbindung zu einem Ihrer sekundären Cluster her. Da es sich bei allen Instanzen um schreibgeschützte Replikatinstanzen handelt, geben Sie eine andere Leseeinstellung als `primary` (z. B., oder) an. `secondary` `secondaryPreferred` `nearest` Zum Minimieren der Latenzzeit wählen Sie denjenigen Leser-Endpunkt aus, der sich in Ihrer -Region oder der Ihnen am nächsten gelegenen -Region befindet.

Überwachung globaler Amazon DocumentDB-Cluster

Amazon DocumentDB (mit MongoDB-Kompatibilität) lässt sich in integrieren, CloudWatch sodass Sie Betriebsmetriken für Ihre Cluster sammeln und analysieren können. Sie können diese Metriken mithilfe der CloudWatch Konsole, der Amazon DocumentDB-Konsole, der AWS Command Line Interface (AWS CLI) oder der API überwachen. CloudWatch

Verwenden Sie die folgenden CloudWatch Metriken, um einen globalen Cluster zu überwachen.

Metrik	Description
`GlobalClusterReplicatedWriteIO`	Die durchschnittliche Anzahl der fakturierten I/O Schreibvorgänge, die vom Cluster-Volume im primären AWS-Region auf das Cluster-Volume in einem sekundären Volume repliziert wurden AWS-Region, wird in 5-Minuten-Intervallen gemeldet. Die Anzahl der Replikationen in `ReplicatedWriteIOs` jede sekundäre Region entspricht der Anzahl der IN-Region

Metrik	Description
	en, die von der primären Region `VolumeWriteIOPs` ausgeführt werden.
`GlobalClusterDataTransferBytes`	Die Datenmenge, die vom primären Cluster auf den eines sekundären Clusters übertragen wird AWS-Region, gemessen in Byte. AWS-Region
`GlobalClusterReplicationLag`	Der Umfang der Verzögerung in Millisekunden, wenn Änderungsereignisse vom primären Cluster auf den eines sekundären Clusters repliziert AWS-Region werden AWS-Region

Weitere Informationen zum Anzeigen dieser Metriken finden Sie unter Daten anzeigen. CloudWatch

Disaster Recovery und globale Amazon DocumentDB-Cluster

Topics

- Durchführung eines verwalteten Failovers für einen globalen Amazon DocumentDB-Cluster
- Durchführung eines manuellen Failovers für einen globalen Amazon DocumentDB-Cluster
- Durchführung eines Switchovers für einen globalen Amazon DocumentDB-Cluster
- Entsperren eines globalen Cluster-Switchovers oder -Failovers
- Verwaltung von RPOs für globale Amazon DocumentDB-Cluster

Mithilfe eines globalen Clusters können Sie sich nach Katastrophen wie regionalen Ausfällen schnell erholen. Die Wiederherstellung nach einem Notfall wird in der Regel anhand von RTO- und RPO-Werten gemessen.

- Recovery Time Objective (RTO) – Die Zeit, die ein System benötigt, um nach einem Notfall in einen arbeitsfähigen Zustand zurückzukehren. Mit anderen Worten: RTO misst die Ausfallzeit. Für einen globalen Cluster: RTO in Minuten.
- Recovery Point Objective (RPO) – Die Datenmenge, die verloren gehen kann (gemessen in Zeit). Bei einem globalen Cluster wird RPO in der Regel in Sekunden gemessen.
- Zur Wiederherstellung nach einem ungeplanten Ausfall können Sie ein regionsübergreifendes Failover zu einem der sekundären Netzwerke in Ihrem globalen Cluster durchführen. Wenn Ihr

globaler Cluster mehrere sekundäre Regionen hat, stellen Sie sicher, dass Sie alle sekundären Regionen trennen, die Sie als primäre Regionen heraufstufen möchten. Anschließend stufen Sie eine dieser sekundären Regionen zur neuen primären Region herauf. AWS-Region Schließlich erstellen Sie neue Cluster in jeder der anderen sekundären Regionen und fügen diese Cluster Ihrem globalen Cluster hinzu.

Durchführung eines verwalteten Failovers für einen globalen Amazon DocumentDB-Cluster

Dieser Ansatz dient der Geschäftskontinuität im Falle einer echten regionalen Katastrophe oder eines kompletten Service-Level-Ausfalls.

Während eines verwalteten Failovers erfolgt ein Failover Ihres primären Clusters auf die sekundäre Region Ihrer Wahl, während die bestehende Replikationstopologie Ihres globalen Amazon DocumentDB-Clusters beibehalten wird. Der gewählte sekundäre Cluster stuft einen seiner schreibgeschützten Knoten auf vollen Writer-Status herauf. In diesem Schritt kann der Cluster die Rolle des primären Clusters übernehmen. Ihre Datenbank ist für kurze Zeit nicht verfügbar, während dieser Cluster seine neue Rolle annimmt. Daten, die nicht vom alten Primärcluster auf den ausgewählten sekundären Cluster repliziert wurden, fehlen möglicherweise, wenn dieser sekundäre Cluster zum neuen primären Cluster wird. Das alte primäre Volume versucht nach besten Kräften, einen Snapshot zu erstellen, bevor es mit dem neuen primären Volume synchronisiert wird, sodass nicht replizierte Daten auf dem Snapshot erhalten bleiben.

> **Note**
>
> Sie können ein verwaltetes regionsübergreifendes Cluster-Failover auf einem globalen Amazon DocumentDB-Cluster nur durchführen, wenn der primäre und alle sekundären Cluster über dieselben Engine-Versionen verfügen. Wenn Ihre Engine-Versionen nicht kompatibel sind, können Sie das Failover manuell durchführen, indem Sie die Schritte unter Durchführung eines manuellen Failovers für einen globalen Amazon DocumentDB-Cluster ausführen.
> Wenn die Engine-Versionen der Region nicht übereinstimmen, wird das Failover blockiert. Suchen Sie nach ausstehenden Upgrades und wenden Sie sie an, um sicherzustellen, dass alle Engine-Versionen der Region übereinstimmen und das globale Cluster-Failover entsperrt ist. Weitere Informationen finden Sie unter Entsperren eines globalen Cluster-Switchovers oder -Failovers.

Gehen Sie wie folgt vor, bevor Sie diese Funktion verwenden, um den Datenverlust zu minimieren:

- Schalten Sie Anwendungen offline, um zu verhindern, dass Schreibvorgänge an den primären Cluster des globalen Amazon DocumentDB-Clusters gesendet werden.
- Überprüfen Sie die Lag-Zeiten für alle sekundären Amazon DocumentDB-Cluster. Durch die Auswahl der sekundären Region mit der geringsten Replikationsverzögerung kann der Datenverlust in der aktuell ausgefallenen primären Region minimiert werden. Überprüfen Sie die Lag-Zeiten für alle sekundären Amazon DocumentDB-Cluster im globalen Cluster, indem Sie sich die `GlobalClusterReplicationLag` Metrik in Amazon ansehen. CloudWatch Diese Metriken zeigen Ihnen, wie weit (in Millisekunden) die Replikation auf einen sekundären Cluster auf den primären Cluster zurückliegt.

  Weitere Informationen zu CloudWatch Metriken für Amazon DocumentDB finden Sie unter. Amazon DocumentDB-Metriken

Während eines verwalteten Failovers wird dem ausgewählten sekundären Cluster seine neue Rolle als primärer Cluster zugewiesen. Er erbt jedoch nicht die verschiedenen Konfigurationsoptionen des primären Clusters. Wenn die Konfiguration nicht übereinstimmt, kann dies Leistungsprobleme, Workload-Inkompatibilitäten und anderes anomales Verhalten zur Folge haben. Um solche Probleme zu vermeiden, beheben Sie die Unterschiede zwischen Ihren globalen Amazon DocumentDB-Clustern für Folgendes:

- Konfigurieren Sie bei Bedarf eine Amazon DocumentDB-Cluster-Parametergruppe für den neuen primären Cluster — Sie können Ihre Amazon DocumentDB-Cluster-Parametergruppen unabhängig für jeden Cluster in Ihrem globalen Amazon DocumentDB-Cluster konfigurieren. Wenn Sie also einen sekundären Cluster so hochstufen, dass er die primäre Rolle übernimmt, wird die Parametergruppe aus dem sekundären Cluster möglicherweise anders konfiguriert als für den primären. Wenn ja, ändern Sie die Parametergruppe des hochgestuften sekundären Clusters so, dass sie den Einstellungen Ihres primären Clusters entspricht. Um zu erfahren wie dies geht, vgl. Ändern der Amazon DocumentDB-Cluster-Parametergruppen.
- Konfigurieren Sie Überwachungstools und -optionen wie CloudWatch Amazon-Ereignisse und -Alarme — Konfigurieren Sie den geförderten Cluster mit denselben Protokollierungsfunktionen, Alarmen usw., wie es für den globalen Cluster erforderlich ist. Wie bei Parametergruppen wird die Konfiguration für diese Funktionen während des Failover-Prozesses nicht vom primären Cluster übernommen. Einige CloudWatch Metriken, wie z. B. die Replikationsverzögerung, sind nur für sekundäre Regionen verfügbar. Daher ändert ein Failover die Art und Weise, wie diese Metriken angezeigt und Alarme für sie eingerichtet werden, und es können Änderungen an allen

vordefinierten Dashboards erforderlich sein. Weitere Informationen zu Amazon DocumentDB-Clustern und zur Überwachung finden Sie unterÜberwachung und Protokollierung in Amazon DocumentDB.

In der Regel übernimmt der gewählte sekundäre Cluster innerhalb einer Minute die primäre Rolle. Sobald der Writer-Knoten der neuen primären Region verfügbar ist, können Sie Ihre Anwendungen mit diesem verbinden und Ihre Workloads fortsetzen. Nachdem Amazon DocumentDB den neuen primären Cluster beworben hat, werden automatisch alle zusätzlichen Cluster der sekundären Region neu erstellt.

Da globale Amazon DocumentDB-Cluster asynchrone Replikation verwenden, kann die Replikationsverzögerung in jeder sekundären Region variieren. Amazon DocumentDB erstellt diese sekundären Regionen neu, sodass sie über exakt dieselben Point-in-Time-Daten verfügen wie der neue Cluster für die primäre Region. Die Dauer der vollständigen Neuerstellungsaufgabe kann je nach Größe des Speichervolumens und der Entfernung zwischen den Regionen einige Minuten bis mehrere Stunden dauern. Wenn der Wiederaufbau der Cluster der sekundären Region aus der neuen primären Region abgeschlossen ist, stehen sie für den Lesezugriff zur Verfügung. Sobald der neue primäre Writer hochgestuft und verfügbar ist, kann der Cluster der neuen primären Region Lese- und Schreibvorgänge für den globalen Amazon DocumentDB-Cluster ausführen.

Um die ursprüngliche Topologie des globalen Clusters wiederherzustellen, überwacht Amazon DocumentDB die Verfügbarkeit der alten primären Region. Sobald diese Region fehlerfrei und wieder verfügbar ist, fügt Amazon DocumentDB sie automatisch als sekundäre Region wieder zum globalen Cluster hinzu. Bevor das neue Speicher-Volume in der alten primären Region erstellt wird, versucht Amazon DocumentDB, einen Snapshot des alten Speicher-Volumes zum Zeitpunkt des Fehlers zu erstellen. Auf diese Weise können Sie alle fehlenden Daten wiederherstellen. Wenn dieser Vorgang erfolgreich ist, platziert Amazon DocumentDB diesen Snapshot mit dem Namen „rds:docdb-unplanned-global-failover-name-of-old-primary-“ im Snapshot-Abschnitt von. DB-cluster-timestamp AWS-Managementkonsole Sie können diesen Snapshot auch in `DescribeDBClusterSnapshots` den Informationen sehen, die von der API-Operation zurückgegeben wurden.

> **Note**
>
> Der Snapshot des alten Speichervolumes ist ein System-Snapshot, der der auf dem alten primären Cluster konfigurierten Aufbewahrungsfrist für Backups unterliegt. Um diesen Snapshot außerhalb des Aufbewahrungszeitraums zu speichern, können Sie ihn kopieren,

um ihn als manuellen Snapshot zu speichern. Weitere Informationen zum Kopieren von Snapshots, einschließlich der Preise, finden Sie unter Einen Cluster-Snapshot kopieren.

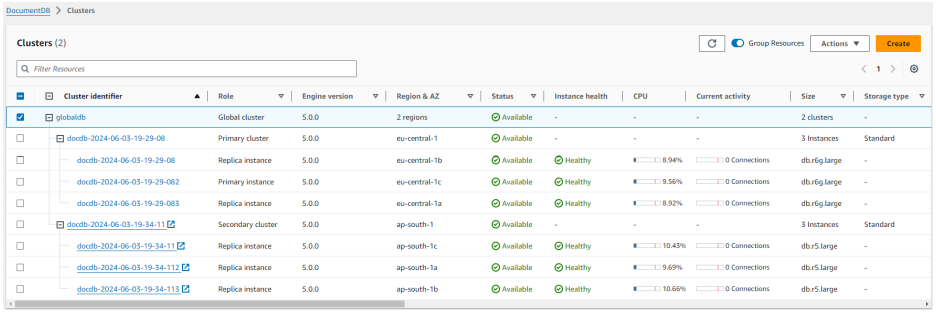
Nach der Wiederherstellung der ursprünglichen Topologie können Sie für Ihren globalen Cluster ein Failback auf die ursprüngliche primäre Region durchführen, indem Sie einen Switchover-Vorgang durchführen, wenn dies für Ihr Unternehmen und Ihre Arbeitslast am sinnvollsten ist. Eine Schritt-für-Schritt-Anleitung hierzu finden Sie unter Durchführung eines Switchovers für einen globalen Amazon DocumentDB-Cluster.

Sie können ein Failover für Ihren globalen Amazon DocumentDB-Cluster mithilfe der AWS-Managementkonsole, der oder der AWS CLI Amazon DocumentDB-API durchführen.

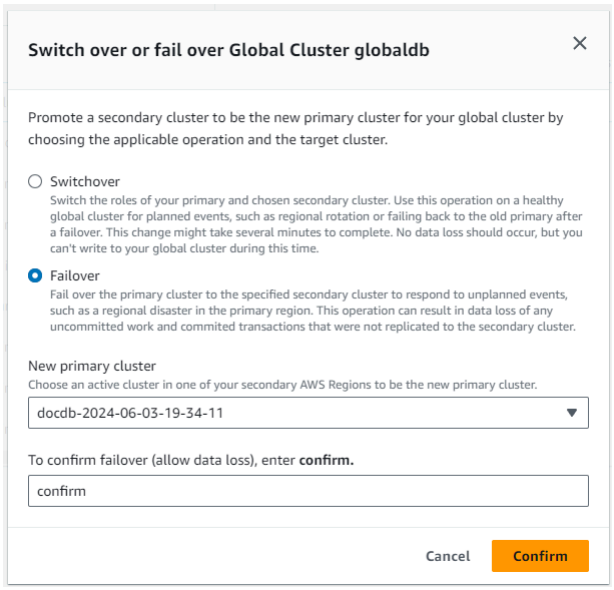
Using the AWS-Managementkonsole

Führen Sie ein verwaltetes Failover auf Ihrem globalen Amazon DocumentDB-Cluster durch

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Klicken Sie im Navigationsbereich auf Cluster.
3. Suchen Sie den globalen Amazon DocumentDB-Cluster, für den Sie ein Failover durchführen möchten, und wählen Sie ihn aus.



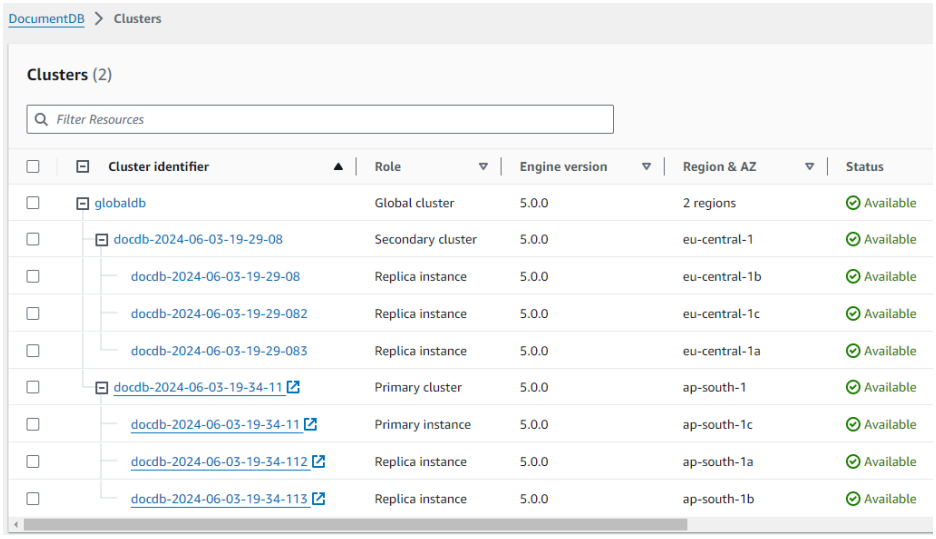
4. Wählen Sie im Menü „Aktionen“ die Option „Switchover“ oder „Failover“.
5. Wählen Sie in dem daraufhin angezeigten Dialogfeld die Option Failover aus und wählen Sie dann den sekundären Cluster aus der Dropdownliste Neues primäres Cluster-Feld aus.



6. Geben Sie in das letzte Feld „Bestätigen“ ein. Wählen Sie dann Confirm (Bestätigen) aus.

   Der Status des primären Clusters ändert sich in "Failing-over“. Dieser Zustand sollte ungefähr eine Minute dauern. Während dieser Zeit lautet der Status des neuen primären Clusters "Wird geändert... “. Sobald der neue Primärserver hochgestuft wird, wird er als "Verfügbar" gekennzeichnet und kann Lese- und Schreibtransaktionen verarbeiten. In den sekundären Regionen, einschließlich der alten primären Region, wird "Resynchronisierung...“ angezeigt "während der Resynchronisierung mit der neuen Primärregion. Ähnlich wie der neue primäre Server kann er Transaktionen erst bearbeiten, wenn sich der Status auf "Verfügbar" ändert.

7. Wenn der Vorgang abgeschlossen ist, wird der ursprüngliche primäre Cluster zum sekundären Cluster. Der ausgewählte sekundäre Cluster wird zum primären Cluster.



Using the AWS CLI

Führen Sie ein verwaltetes Failover auf Ihrem globalen Amazon DocumentDB-Cluster durch

Führen Sie den `failover-global-cluster` CLI-Befehl aus, um ein Failover für Ihren globalen Amazon DocumentDB-Cluster durchzuführen. Übergeben Sie mit dem Befehl Werte für die folgenden Optionen:

- `--region`
- `--global-cluster-identifier`
- `--target-db-cluster-identifier`
- `--allow-data-loss`

Ersetzen Sie in den folgenden Beispielen jedes *`user input placeholder`* durch die Informationen Ihres Clusters.

Für Linux, macOS oder Unix:

```
aws docdb failover-global-cluster \
   --region region_of_selected_secondary \
   --global-cluster-identifier global_cluster_id \
   --target-db-cluster-identifier arn_of_secondary_to_promote \
   --allow-data-loss
```

Für Windows:

```
aws docdb failover-global-cluster ^
   --region region_of_selected_secondary ^
   --global-cluster-identifier global_cluster_id ^
   --target-db-cluster-identifier arn_of_secondary_to_promote ^
   --allow-data-loss
```

Durchführung eines manuellen Failovers für einen globalen Amazon DocumentDB-Cluster

Wenn ein ganzer Cluster in einem Cluster nicht AWS-Region mehr verfügbar ist, können Sie einen anderen Cluster im globalen Cluster so hochstufen, dass er über read/write die entsprechenden Funktionen verfügt.

Sie können den globalen Cluster-Failover-Mechanismus manuell aktivieren, wenn ein Cluster in einem anderen Cluster die bessere Wahl als primärer Cluster AWS-Region ist. Sie können beispielsweise die Kapazität eines sekundären Clusters erhöhen und diesen Cluster dann zum primären Cluster hochstufen. Oder das Aktivitätsgleichgewicht zwischen den Clustern AWS-Regionen könnte sich ändern, sodass das Umschalten des primären Clusters auf einen anderen zu einer geringeren Latenz bei Schreibvorgängen führen AWS-Region kann.

Das folgende Verfahren beschreibt, wie Sie einen der sekundären Cluster in einem globalen Amazon DocumentDB-Cluster fördern können.

So fördern Sie einen sekundären Cluster:

1. Beenden Sie die Ausgabe von DML-Anweisungen und anderen Schreibvorgängen an den primären Cluster während des AWS-Region Ausfalls.
2. Identifizieren Sie einen Cluster aus einem sekundären Cluster AWS-Region , der als neuer primärer Cluster verwendet werden soll. Wenn Sie zwei (oder mehr) sekundäre Cluster AWS-Regionen in Ihrem globalen Cluster haben, wählen Sie den sekundären Cluster aus, der die geringste Verzögerung aufweist.
3. Trennen Sie den ausgewählten sekundären Cluster vom globalen Cluster.

   Wenn Sie einen sekundären Cluster aus einem globalen Cluster entfernen, wird die Replikation vom primären Cluster auf diesen sekundären Cluster sofort beendet und der Cluster wird zu einem eigenständigen, bereitgestellten Cluster mit vollem read/write Funktionsumfang heraufgestuft. Alle anderen sekundären Cluster, die dem primären Cluster in der Region zugeordnet sind, in der der Ausfall aufgetreten ist, sind weiterhin verfügbar und können Anrufe von Ihrer Anwendung annehmen. Sie verbrauchen auch Ressourcen. Da Sie den globalen Cluster neu erstellen, entfernen Sie in den folgenden Schritten die anderen sekundären Cluster, bevor Sie den neuen globalen Cluster erstellen, um Split-Brain- und andere Probleme zu vermeiden.

   Ausführliche Schritte zum Trennen finden Sie unter Einen Cluster aus einem globalen Amazon DocumentDB-Cluster entfernen.
4. Dieser Cluster wird zum primären Cluster eines neuen globalen Clusters, wenn Sie im nächsten Schritt beginnen, ihm Regionen hinzuzufügen.
5. Fügen Sie eine AWS-Region zum Cluster hinzu. Wenn Sie dies tun, beginnt der Replikationsprozess vom primären zum sekundären Cluster.

6. Fügen Sie AWS-Regionen bei Bedarf weitere hinzu, um die zur Unterstützung Ihrer Anwendung erforderliche Topologie neu zu erstellen. Stellen Sie sicher, dass Anwendungsschreibvorgänge vor, während und nach solchen Änderungen an den richtigen Cluster gesendet werden, um Dateninkonsistenzen zwischen den Clustern im globalen Cluster zu vermeiden (Split-Brain-Probleme).
7. Wenn der Ausfall behoben ist und Sie bereit sind, Ihren ursprünglichen Cluster erneut AWS-Region als primären Cluster zuzuweisen, führen Sie dieselben Schritte in umgekehrter Reihenfolge aus.
8. Entfernen Sie einen der sekundären Cluster aus dem globalen Cluster. Dadurch kann der read/write Datenverkehr bedient werden.
9. Leiten Sie den gesamten Schreibverkehr auf den primären Cluster im Original um AWS-Region.
10. Fügen Sie eine hinzu AWS-Region , um einen oder mehrere sekundäre Cluster auf die gleiche Weise AWS-Region wie zuvor einzurichten.

Globale Amazon DocumentDB-Cluster können mithilfe von AWS SDKs verwaltet werden, sodass Sie Lösungen zur Automatisierung des globalen Cluster-Failover-Prozesses für Disaster Recovery und Business Continuity Planning erstellen können. Eine solche Lösung wird unseren Kunden unter der Apache 2.0-Lizenz zur Verfügung gestellt und kann in unserem Tool-Repository hier abgerufen werden. https://github.com/awslabs/amazon-documentdb-tools/tree/master/global-clusters-automation Diese Lösung nutzt Amazon Route 53 für das Endpunktmanagement und bietet AWS Lambda Funktionen, die bei entsprechenden Ereignissen ausgelöst werden können.

Durchführung eines Switchovers für einen globalen Amazon DocumentDB-Cluster

Mithilfe von Umstellungen können Sie die Region Ihres primären Clusters routinemäßig ändern. Dieser Ansatz ist für kontrollierte Szenarien gedacht, z. B. für betriebliche Wartungen und andere geplante Betriebsverfahren.

Es gibt drei gängige Anwendungsfälle für die Verwendung von Switchovers:

- Für Anforderungen an die „regionale Rotation“, die in bestimmten Branchen gelten. Beispielsweise könnten die Vorschriften für Finanzdienstleistungen vorschreiben, dass Tier-0-Systeme für mehrere Monate in eine andere Region wechseln müssen, um sicherzustellen, dass die Notfallwiederherstellungsverfahren regelmäßig geübt werden.
- Für regionsübergreifende „Follow-the-Sun“-Anwendungen. Beispielsweise möchte ein Unternehmen möglicherweise Schreibvorgänge mit niedrigerer Latenz in verschiedenen Regionen bereitstellen, basierend auf den Geschäftszeiten in verschiedenen Zeitzonen.

- Als Methode ohne Datenverlust, um nach einem Failover zur ursprünglichen Primärregion zurückzukehren.

> **Note**
>
> Switchovers sind für den Einsatz in einem gesunden globalen Amazon DocumentDB-Cluster konzipiert. Folgen Sie zur Wiederherstellung nach einem ungeplanten Ausfall dem entsprechenden Verfahren unter Durchführung eines manuellen Failovers für einen globalen Amazon DocumentDB-Cluster.
> Um einen Switchover durchzuführen, müssen alle sekundären Regionen exakt dieselbe Engine-Version wie die primäre verwenden. Wenn die Engine-Versionen der Region nicht übereinstimmen, wird der Switchover blockiert. Suchen Sie nach ausstehenden Upgrades und wenden Sie sie an, um sicherzustellen, dass alle Engine-Versionen der Region übereinstimmen und der globale Cluster-Switchover entsperrt wird. Weitere Informationen finden Sie unter Entsperren eines globalen Cluster-Switchovers oder -Failovers.

Während eines Switchovers schaltet Amazon DocumentDB Ihren primären Cluster auf die von Ihnen gewählte sekundäre Region um, während die bestehende Replikationstopologie Ihres globalen Clusters beibehalten wird. Bevor Amazon DocumentDB mit dem Switchover-Prozess beginnt, wartet es, bis alle Cluster der sekundären Region vollständig mit dem Cluster der primären Region synchronisiert sind. Dann wird der DB-Cluster in der primären Region schreibgeschützt, und der ausgewählte sekundäre Cluster stuft einen seiner schreibgeschützten Knoten auf vollen Writer-Status hoch. Durch die Heraufstufung dieses Knotens zu einem Writer kann dieser sekundäre Cluster die Rolle des primären Clusters übernehmen. Da alle sekundären Cluster zu Beginn des Prozesses mit dem primären Cluster synchronisiert wurden, setzt der neue primäre Cluster den Betrieb für den globalen Amazon DocumentDB-Cluster fort, ohne dass Daten verloren gehen. Ihre Datenbank ist für kurze Zeit nicht verfügbar, während der primäre und der ausgewählte sekundäre Cluster ihre neuen Rollen übernehmen.

Gehen Sie wie folgt vor, bevor Sie diese Funktion verwenden, um die Anwendungsverfügbarkeit zu optimieren:

- Führen Sie diesen Vorgang außerhalb der Spitzenzeiten oder zu einem anderen Zeitpunkt durch, zu dem nur wenige Schreibvorgänge in den primären Cluster erfolgen.
- Schalten Sie Anwendungen offline, um zu verhindern, dass Schreibvorgänge an den primären Cluster des globalen Amazon DocumentDB-Clusters gesendet werden.

- Überprüfen Sie die Lag-Zeiten für alle sekundären Amazon DocumentDB-Cluster im globalen Cluster, indem Sie sich die `GlobalClusterReplicationLag` Metrik in Amazon ansehen. CloudWatch Diese Metrik zeigt Ihnen, wie weit (in Millisekunden) die Replikation auf einen sekundären Cluster auf den primären Cluster zurückliegt. Dieser Wert ist direkt proportional zu der Zeit, die Amazon DocumentDB benötigt, um den Switchover abzuschließen. Je größer der Verzögerungswert, desto länger dauert die Umstellung.

  Weitere Informationen zu CloudWatch Metriken für Amazon DocumentDB finden Sie unter. Amazon DocumentDB-Metriken

Während einer Umstellung wird der ausgewählte sekundäre DB-Cluster in seine neue Rolle als primärer Cluster hochgestuft. Er übernimmt jedoch nicht die verschiedenen Konfigurationsoptionen des primären DB-Clusters. Wenn die Konfiguration nicht übereinstimmt, kann dies Leistungsprobleme, Workload-Inkompatibilitäten und anderes anomales Verhalten zur Folge haben. Um solche Probleme zu vermeiden, beheben Sie die Unterschiede zwischen Ihren globalen Amazon DocumentDB-Clustern für Folgendes:

- Konfigurieren Sie bei Bedarf die Amazon DocumentDB-DB-Cluster-Parametergruppe für den neuen primären Cluster — Sie können Ihre Amazon DocumentDB-Cluster-Parametergruppen unabhängig für jeden Cluster in Ihrem globalen Amazon DocumentDB-Cluster konfigurieren. Wenn Sie also einen sekundären DB-Cluster zur Übernahme der primären Rolle hochstufen, kann die Parametergruppe des sekundären Clusters im Vergleich zum primären Cluster möglicherweise anders konfiguriert werden. Ist dies der Fall, ändern Sie die Parametergruppe des hochgestuften sekundären DB-Clusters so, dass sie den Einstellungen des primären Clusters entspricht. Um zu erfahren wie, siehe Verwaltung von Amazon DocumentDB-Cluster-Parametergruppen.
- Konfigurieren Sie Überwachungstools und -optionen wie CloudWatch Amazon-Ereignisse und -Alarme — Konfigurieren Sie den beworbenen Cluster mit denselben Protokollierungsfunktionen, Alarmen usw., je nach Bedarf für den globalen Cluster. Wie bei Parametergruppen wird die Konfiguration für diese Funktionen während der Umstellung nicht vom primären Cluster übernommen. Einige CloudWatch Metriken, wie z. B. die Replikationsverzögerung, sind nur für primäre Regionen verfügbar. Daher ändert sich bei einer Umstellung die Art und Weise, wie diese Metriken angezeigt und Alarme für sie eingerichtet werden, und es können Änderungen an allen vordefinierten Dashboards erforderlich sein. Weitere Informationen finden Sie unter Überwachung und Protokollierung in Amazon DocumentDB.

> **Note**
>
> In der Regel kann der Rollenwechsel bis zu mehreren Minuten dauern.

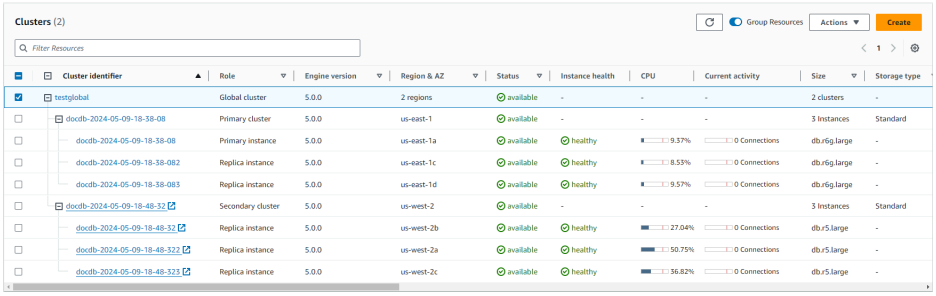
Wenn der Switchover-Prozess abgeschlossen ist, kann der beworbene Amazon DocumentDB-Cluster Schreibvorgänge für den globalen Cluster verarbeiten.

Sie können Ihren globalen Amazon DocumentDB-Cluster mit dem oder dem AWS-Managementkonsole : AWS CLI

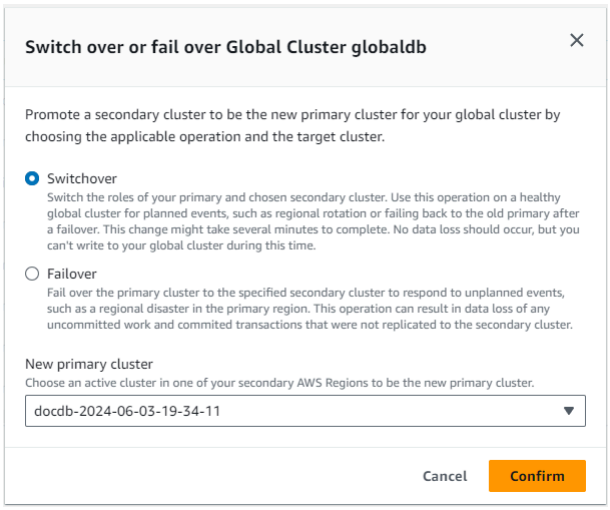
Using the AWS-Managementkonsole

Führen Sie einen Switchover auf Ihrem globalen Amazon DocumentDB-Cluster durch

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Klicken Sie im Navigationsbereich auf Cluster.
3. Suchen Sie den globalen Amazon DocumentDB-Cluster, den Sie wechseln möchten, und wählen Sie ihn aus.



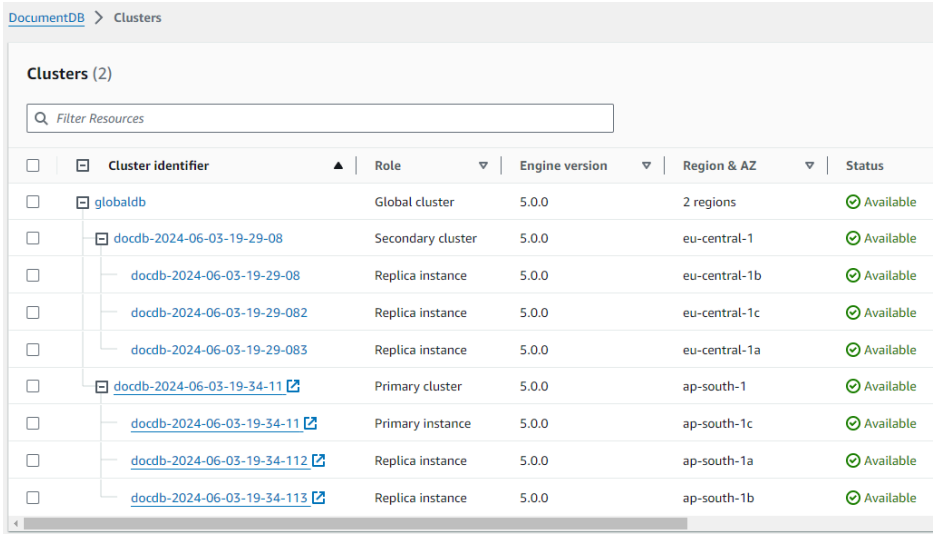
4. Wählen Sie im Menü „Aktionen“ die Option „Switchover“ oder „Failover“.
5. Wählen Sie in dem daraufhin angezeigten Dialogfeld Switchover aus und wählen Sie dann den sekundären Cluster aus der Dropdownliste Neues primäres Cluster-Feld aus.



6. Wählen Sie Bestätigen aus.

   Der Status des primären Clusters ändert sich in "Switching-over“. Dieser Zustand sollte ungefähr drei Minuten dauern. Während dieser Zeit lautet der Status aller regionalen Cluster "Wird geändert... “. Sobald die Regionen synchronisiert sind und die neue primäre Region hochgestuft wurde, wird in allen Statusfeldern der Wert "Verfügbar" angezeigt und Transaktionen können bearbeitet werden.

7. Wenn der Vorgang abgeschlossen ist, wird der ursprüngliche primäre Cluster zum sekundären Cluster. Der ausgewählte sekundäre Cluster wird zum primären Cluster.



Using the AWS CLI

Führen Sie einen Switchover auf Ihrem globalen Amazon DocumentDB-Cluster durch

Führen Sie den `switchover-global-cluster` CLI-Befehl aus, um Ihren globalen Amazon DocumentDB-Cluster umzuschalten. Übergeben Sie mit dem Befehl Werte für die folgenden Optionen:

- `--region`
- `--global-cluster-identifier`
- `--target-db-cluster-identifier`

Ersetzen Sie in den folgenden Beispielen jedes *`user input placeholder`* durch die Informationen Ihres Clusters.

Für Linux, macOS oder Unix:

```
aws docdb switchover-global-cluster \
   --region region_of_primary \
   --global-cluster-identifier global_cluster_id \
   --target-db-cluster-identifier arn_of_secondary_to_promote
```

Für Windows:

```
aws docdb switchover-global-cluster ^
   --region region_of_primary ^
   --global-cluster-identifier global_cluster_id ^
   --target-db-cluster-identifier arn_of_secondary_to_promote
```

Entsperren eines globalen Cluster-Switchovers oder -Failovers

Regionen mit unterschiedlichen Engine-Versionen

Globale Cluster-Switchover und Failover werden blockiert, wenn nicht alle regionalen Cluster im globalen Cluster dieselbe Engine-Version verwenden. Wenn die Versionen nicht übereinstimmen, wird möglicherweise dieser Fehler angezeigt, wenn Sie einen Switchover oder Failover aufrufen: Auf dem angegebenen Ziel-DB-Cluster wird eine Engine-Version mit einem anderen Patch-Level ausgeführt als auf dem Quell-DB-Cluster. Wenden Sie routinemäßig die neuesten Engine-Versionen an, um Ihre globalen Cluster in einem einwandfreien Zustand zu halten.

Um diesen Fehler zu beheben, aktualisieren Sie zuerst alle sekundären Regionen und dann die primäre Region auf dieselbe Engine-Version, indem Sie alle ausstehenden Wartungsmaßnahmen

anwenden. Folgen Sie den Anweisungen auf einer der folgenden Registerkarten, um ausstehende Wartungsmaßnahmen anzuzeigen und die zur Behebung des Problems erforderlichen Änderungen vorzunehmen:

Using the AWS-Managementkonsole

Um die Blockierung eines globalen Cluster-Switchovers oder -Failovers aufzuheben, müssen Sie ermitteln, ob Wartungsmaßnahmen für Ihre Cluster ausstehen, und diese anwenden. Gehen Sie wie folgt vor, um Wartungsmaßnahmen anzuzeigen und anzuwenden:

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Klicken Sie im Navigationsbereich auf Cluster.
3. Suchen Sie in der Cluster-Tabelle in der Spalte Cluster-ID nach Ihrem globalen Cluster. Notieren Sie sich unter Ihrem globalen Cluster jeden sekundären Cluster und den primären Cluster für den angegebenen globalen Cluster und führen Sie für jeden Cluster die folgenden Schritte aus.
4. Für jeden sekundären Cluster:
   a. Wenn ein Update für Ihren Cluster verfügbar ist, wird es in der Spalte Wartung als Verfügbar , Erforderlich oder Nächstes Fenster angezeigt.
   b. Um eine Aktion auszuführen, wählen Sie den Cluster aus, um dessen Details anzuzeigen, und wählen Sie dann Wartung und Backups aus. Die Elemente „Ausstehende Wartung“ werden angezeigt.
   c. Wenn unter Beschreibung angezeigt wird, dass ein „Neues Wartungsupdate verfügbar ist“, wählen Sie es aus und klicken Sie dann auf Jetzt anwenden.
5. Für Ihren primären Cluster:
   a. Wenn ein Update für Ihren Cluster verfügbar ist, wird es in der Spalte Wartung als Verfügbar , Erforderlich oder Nächstes Fenster angezeigt.
   b. Um eine Aktion auszuführen, wählen Sie den Cluster aus, um dessen Details anzuzeigen, und wählen Sie dann Wartung und Backups aus. Die Elemente „Ausstehende Wartung“ werden angezeigt.
   c. Wenn unter Beschreibung angezeigt wird, dass ein „Neues Wartungsupdate verfügbar ist“, wählen Sie es aus und klicken Sie dann auf Jetzt anwenden.

Using the AWS CLI

Um die Blockierung eines globalen Cluster-Switchovers oder -Failovers aufzuheben, müssen Sie ermitteln, ob Wartungsmaßnahmen für den Cluster ausstehen, und diese anwenden. Gehen Sie wie folgt vor, um Wartungsmaßnahmen zuerst auf den sekundären Clustern und dann auf dem primären Cluster für Ihren globalen Cluster anzuzeigen und anzuwenden:

1. Führen Sie die folgenden Schritte zuerst für den regionalen Cluster jeder sekundären Region und dann für den regionalen Cluster der primären Regionen aus.
2. Führen Sie den `describe-pending-maintenance-actions` CLI-Befehl mit der `--resource-identifier` Option aus, um festzustellen, ob Wartungsmaßnahmen für Ihren regionalen Amazon DocumentDB-Cluster verfügbar sind.

   Ersetzen Sie in den folgenden Beispielen jedes *`user input placeholder`* durch die Informationen Ihres Clusters.

   Für Linux, macOS oder Unix:

   ```
   aws docdb describe-pending-maintenance-action \
      --resource-identifier arn:aws:rds:us-
   east-1:001234567890:cluster:docdb-2025-03-27-19-21-15
   ```

   Für Windows:

   ```
   aws docdb describe-pending-maintenance-action ^
      --resource-identifier arn:aws:rds:us-
   east-1:001234567890:cluster:docdb-2025-03-27-19-21-15
   ```

   Das Ergebnis sieht ungefähr so aus:

   ```
   {
       "PendingMaintenanceActions": [
           {
               "ResourceIdentifier": "arn:aws:rds:us-
   east-1:001234567890:cluster:docdb-2025-03-27-19-21-15",
               "PendingMaintenanceActionDetails": [
                   {
                       "Action": "system-update",
                       "CurrentApplyDate": "2025-04-11T03:01:00Z",
                       "Description": "db-version-upgrade",
   ```

```
                    "ForcedApplyDate": "2025-06-18T03:01:00Z",
                    "AutoAppliedAfterDate": "2025-05-11T03:01:00Z"
                    "OptInStatus": "pending"
                }
            ]
        }
    ]
}
```

3. Wenn eine Wartungsaktion erforderlich ist, führen Sie den [apply-pending-maintenance-action](#) CLI-Befehl mit den folgenden Optionen aus:

   - `--resource-identifier`
   - `--apply-action`
   - `--opt-in-type`
   - `--region`

   Ersetzen Sie in den folgenden Beispielen jedes *`user input placeholder`* durch die Informationen Ihres Clusters.

   Für Linux, macOS oder Unix:

```
aws docdb apply-pending-maintenance-action \
   --resource-identifier arn:aws:rds:us-
east-1:001234567890:cluster:docdb-2025-03-27-19-21-15 \
   --apply-action system-update \
   --opt-in-type immediate \
   --region us-east-1
```

   Für Windows:

```
aws docdb apply-pending-maintenance-action ^
   --resource-identifier arn:aws:rds:us-
east-1:001234567890:cluster:docdb-2025-03-27-19-21-15 ^
   --apply-action system-update ^
   --opt-in-type immediate ^
   --region us-east-1
```

4. Sobald die Wartungsaktion abgeschlossen ist, führen Sie den `describe-pending-maintenance-actions` Befehl erneut aus, um sicherzustellen, dass keine weiteren Aktionen für Ihren Cluster ausstehen.

   Das gewünschte Ergebnis ist:

```
{
    "PendingMaintenanceActions": []
}
```

Using the Amazon DocumentDB API

Um die Blockierung eines globalen Cluster-Switchovers oder -Failovers aufzuheben, müssen Sie ermitteln, ob Wartungsmaßnahmen für den Cluster ausstehen, und diese anwenden. Verwenden Sie die folgenden APIs, um Wartungsmaßnahmen anzuzeigen und anzuwenden:

1. Führen Sie die folgenden Schritte zuerst für den regionalen Cluster jeder sekundären Region und dann für den regionalen Cluster der primären Regionen aus.
2. Rufen Sie die PendingMaintenanceAction API auf, um festzustellen, ob Wartungsmaßnahmen für Ihren globalen Amazon DocumentDB-Cluster verfügbar sind.
3. Wenden Sie alle Änderungen an, indem Sie die ApplyPendingMaintenanceAction API aufrufen.

Ausstehende Änderungen am Zielcluster

Switchover und Failover werden auch blockiert, wenn für den sekundären Zielcluster eine geplante Änderung vorgenommen wurde, die noch nicht angewendet wurde, z. B. eine Änderung oder Wartungsmaßnahme, die Sie für das nächste Wartungsfenster angefordert haben. In diesem Fall wird möglicherweise dieser Fehler angezeigt, wenn Sie einen Switchover oder Failover aufrufen: Sie können kein Failover zum Cluster mit ARN durchführen, *`arn:aws:rds:us-east-1:001234567890:cluster:docdb-2025-03-27-19-21-15`* weil er gerade geändert wird oder weil Änderungen ausstehen. Versuchen Sie es erneut, wenn der Cluster verfügbar ist. Um den Vorgang zu entsperren, entfernen Sie die geplante Änderung auf dem Zielcluster. Die Schritte hängen von der Art der Änderung ab. Nachdem die geplante Änderung entfernt wurde und der Cluster-Status wieder erreicht ist, versuchen Sie erneut`available`, den Switchover oder Failover durchzuführen.

Brechen Sie eine geplante Wartungsaktion ab

Sie können eine geplante Wartungsaktion (Rückgängigmachen einer Anmeldung) nicht von der AWS-Managementkonsole aus stornieren. Verwenden Sie daher die AWS CLI.

Using the AWS CLI

1. Führen Sie den CLI-Befehl https://awscli.amazonaws.com/v2/documentation/api/latest/reference/docdb/describe-pending-maintenance-actions.html describe-pending-maintenance-actions mit der `--resource-identifier` Option aus und suchen Sie nach einer Wartungsaktion, deren Name lautet. `OptInStatus` `next-maintenance`

   Ersetzen Sie in den folgenden Beispielen jedes *`user input placeholder`* durch die Informationen Ihres Clusters.

   Für Linux, macOS oder Unix:

   ```
   aws docdb describe-pending-maintenance-actions \
      --resource-identifier arn:aws:rds:us-
   east-1:001234567890:cluster:docdb-2025-03-27-19-21-15 \
      --region us-east-1
   ```

   Für Windows:

   ```
   aws docdb describe-pending-maintenance-actions ^
      --resource-identifier arn:aws:rds:us-
   east-1:001234567890:cluster:docdb-2025-03-27-19-21-15 ^
      --region us-east-1
   ```

   Das Ergebnis sieht ähnlich wie das Folgende aus. Notieren Sie sich den `Action` Wert (`os-upgrade`in diesem Beispiel), den Sie im nächsten Schritt verwenden.

   ```
   {
       "PendingMaintenanceActions": [
           {
               "ResourceIdentifier": "arn:aws:rds:us-
   east-1:001234567890:cluster:docdb-2025-03-27-19-21-15",
               "PendingMaintenanceActionDetails": [
                   {
                       "Action": "os-upgrade",
                       "OptInStatus": "next-maintenance",
   ```

```
                    "CurrentApplyDate": "2026-09-02T03:02:00Z",
                    "Description": "New Operating System update is available"
                }
            ]
        }
    ]
}
```

2. Brechen Sie die geplante Aktion ab, indem Sie den CLI-Befehl apply-pending-maintenance-action mit ausführen und den `Action` Wert aus dem `--opt-in-type undo-opt-in` vorherigen Schritt an übergeben. `--apply-action`

   Für Linux, macOS oder Unix:

```
aws docdb apply-pending-maintenance-action \
   --resource-identifier arn:aws:rds:us-
east-1:001234567890:cluster:docdb-2025-03-27-19-21-15 \
   --apply-action os-upgrade \
   --opt-in-type undo-opt-in \
   --region us-east-1
```

   Für Windows:

```
aws docdb apply-pending-maintenance-action ^
   --resource-identifier arn:aws:rds:us-
east-1:001234567890:cluster:docdb-2025-03-27-19-21-15 ^
   --apply-action os-upgrade ^
   --opt-in-type undo-opt-in ^
   --region us-east-1
```

   In der Antwort hat die Wartungsaktion keine mehr, was bestätigt, dass das geplante Opt-In `OptInStatus` storniert wurde.

```
{
    "ResourcePendingMaintenanceActions": {
        "ResourceIdentifier": "arn:aws:rds:us-
east-1:001234567890:cluster:docdb-2025-03-27-19-21-15",
        "PendingMaintenanceActionDetails": [
            {
                "Action": "os-upgrade",
                "Description": "New Operating System update is available"
```

```
            }
        ]
    }
}
```

Macht eine geplante Cluster-Änderung rückgängig

Using the AWS CLI

1. Suchen Sie nach geplanten Änderungen, indem Sie den CLI-Befehl https://awscli.amazonaws.com/v2/documentation/api/latest/reference/rds/describe-db-clusters.html describe-db-clusters ausführen und das Ausgabefeld überprüfen. `PendingModifiedValues`

   > **Note**
   >
   > Verwenden Sie für diesen Befehl die Amazon RDS CLI (`aws rds`), nicht die Amazon DocumentDB CLI (`aws docdb`), da die Amazon `describe-db-clusters` DocumentDB-API kein Feld auf Clusterebene zurückgibt. `PendingModifiedValues`

   Ersetzen Sie in den folgenden Beispielen jedes durch die Informationen Ihres *`user input placeholder`* Clusters.

   Für Linux, macOS oder Unix:

   ```
   aws rds describe-db-clusters \
      --db-cluster-identifier docdb-2025-03-27-19-21-15 \
      --query 'DBClusters[0].PendingModifiedValues' \
      --region us-east-1
   ```

   Für Windows:

   ```
   aws rds describe-db-clusters ^
      --db-cluster-identifier docdb-2025-03-27-19-21-15 ^
      --query "DBClusters[0].PendingModifiedValues" ^
      --region us-east-1
   ```

Das Ergebnis zeigt die geplanten Änderungen. In diesem Beispiel steht eine Änderung des Clusternamens (`DBClusterIdentifier`) aus, und zwar von `docdb-2025-03-27-19-21-15` bis`docdb-new-name`.

```
{
    "DBClusterIdentifier": "docdb-new-name"
}
```

2. Machen Sie die Änderung rückgängig, indem Sie den CLI-Befehl modify-db-cluster mit `--apply-immediately` ausführen und den geänderten Wert auf seinen ursprünglichen Wert zurücksetzen. In diesem Beispiel wird die ausstehende Namensänderung rückgängig gemacht, indem es auf den aktuellen Namen des Clusters `--new-db-cluster-identifier` zurückgesetzt wird.

   Für Linux, macOS oder Unix:

```
aws docdb modify-db-cluster \
   --db-cluster-identifier docdb-2025-03-27-19-21-15 \
   --new-db-cluster-identifier docdb-2025-03-27-19-21-15 \
   --apply-immediately \
   --region us-east-1
```

   Für Windows:

```
aws docdb modify-db-cluster ^
   --db-cluster-identifier docdb-2025-03-27-19-21-15 ^
   --new-db-cluster-identifier docdb-2025-03-27-19-21-15 ^
   --apply-immediately ^
   --region us-east-1
```

Verwaltung von RPOs für globale Amazon DocumentDB-Cluster

Mit einem globalen Amazon DocumentDB-Cluster können Sie das Recovery Point Objective (RPO) mithilfe des Parameters verwalten. `global_db_rpo` RPO gibt die maximale Datenmenge an, die im Falle eines Ausfalls verloren gehen kann.

Wenn Sie ein RPO für Ihren globalen Amazon DocumentDB-Cluster festlegen, überwacht Amazon DocumentDB die RPO-Lag-Zeit aller sekundären Cluster. Diese Überwachung stellt sicher, dass mindestens ein sekundärer Cluster innerhalb des Ziel-RPO-Fensters bleibt.

Die RPO-Einstellung steuert, wie Amazon DocumentDB Schreibtransaktionen auf dem primären Cluster verwaltet, um potenziellen Datenverlust im Falle eines Failovers zu begrenzen. Amazon DocumentDB bewertet RPO- und RPO-Lagzeiten, um Transaktionen auf dem Primärserver wie folgt festzuschreiben (oder zu blockieren):

- Die Transaktion wird durchgeführt, wenn mindestens ein sekundärer DB-Cluster eine RPO-Verzögerungszeit hat, die kleiner ist als die RPO.
- Die Transaktion wird blockiert, wenn alle sekundären DB-Cluster RPO-Verzögerungszeiten haben, die größer sind als die RPO.

Mit anderen Worten, wenn sich alle sekundären Cluster hinter dem Ziel-RPO befinden, unterbricht Amazon DocumentDB Transaktionen auf dem primären Cluster. Amazon DocumentDB nimmt die unterbrochenen Transaktionen wieder auf und schreibt sie fest, sobald die Lag-Zeit von mindestens einem sekundären DB-Cluster unter das RPO fällt. Das Ergebnis ist, dass keine Transaktionen übertragen werden können, bis die RPO erreicht ist.

Der `global_db_rpo`-Parameter ist dynamisch. Wenn Sie nicht möchten, dass alle Schreibtransaktionen angehalten werden, bis die Verzögerung ausreichend abnimmt, können Sie ihn schnell zurücksetzen. In diesem Fall wendet Amazon DocumentDB die Änderung nach einer kurzen Verzögerung an.

> **Important**
>
> In einer globalen Datenbank mit nur zwei AWS Regionen empfehlen wir, den Standardwert des `global_db_rpo` Parameters in der Parametergruppe der sekundären Region beizubehalten. Andernfalls könnte die Durchführung eines Failovers aufgrund eines Verlusts der primären AWS Region dazu führen, dass Amazon DocumentDB Transaktionen unterbricht. Warten Sie stattdessen, bis Amazon DocumentDB die Neuerstellung des Clusters in der alten ausgefallenen AWS Region abgeschlossen hat, bevor Sie diesen Parameter ändern, um ein maximales RPO durchzusetzen.

Themen

- Festlegen des Recovery Point Objective

- Anzeigen des Recovery Point Objective (Zeitraums zwischen zwei Datensicherungen)
- Deaktivieren des Recovery Point Objective

Festlegen des Recovery Point Objective

Der `global_db_rpo` Parameter steuert die RPO-Einstellung für eine Amazon DocumentDB-Datenbank. Gültige Werte reichen von 20 Sekunden bis 2.147.483.647 Sekunden (68 Jahre). Wählen Sie einen realistischen Wert, der Ihren Geschäftsanforderungen entspricht. Wenn Sie beispielsweise bis zu 10 Minuten für Ihre RPO festlegen möchten, dann setzen Sie den Wert auf 600.

Sie können diesen Wert für Ihren globalen Amazon DocumentDB-Cluster mithilfe der AWS-Managementkonsole, der oder der AWS CLI Amazon DocumentDB-API festlegen.

Using the AWS-Managementkonsole

So legen Sie den RPO fest:

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Wählen Sie den primären Cluster Ihres globalen Amazon DocumentDB-Clusters und öffnen Sie die Registerkarte Konfiguration, um die zugehörige DB-Cluster-Parametergruppe zu finden.

   Parametergruppen können nicht direkt bearbeitet werden. Stattdessen führen Sie die folgenden Schritte aus:

   - Erstellen Sie eine benutzerdefinierte DB-Cluster-Parametergruppe, wobei Sie die entsprechende Standardparametergruppe als Ausgangspunkt verwenden.
   - Stellen Sie in Ihrer benutzerdefinierten DB-Cluster-Parametergruppe den Wert des Parameters global_db_rpo so ein, dass er Ihrem Anwendungsfall entspricht. Gültige Werte liegen zwischen 20 Sekunden und dem maximalen ganzzahligen Wert von 2.147.483.647 (68 Jahre).
   - Wenden Sie die geänderte DB-Cluster-Parametergruppe auf Ihren Amazon DocumentDB-DB-Cluster an.

Weitere Informationen zum Ändern von DB-Cluster-Parametergruppen finden Sie unterÄndern der Amazon DocumentDB-Cluster-Parametergruppen.

Using the AWS CLI

Zum Festlegen des Parameters `global_db_rpo` verwenden Sie den CLI-Befehl modify-db-cluster-parameter-group. Geben Sie in dem Befehl den Namen der Parametergruppe Ihres primären Clusters und Werte für den RPO-Parameter an.

Im folgenden Beispiel wird die RPO für die primäre DB-Cluster-Parametergruppe namens auf 600 Sekunden (10 Minuten) festgeleg `my_custom_global_parameter_group`.

Für Linux, macOS oder Unix:

```
aws docdb modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name my_custom_global_parameter_group \
    --parameters
 "ParameterName=global_db_rpo,ParameterValue=600,ApplyMethod=immediate"
```

Für Windows:

```
aws docdb modify-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name my_custom_global_parameter_group ^
    --parameters
 "ParameterName=global_db_rpo,ParameterValue=600,ApplyMethod=immediate"
```

Using the Amazon DocumentDB API

Verwenden Sie die ModifyDBClusterParameterGroup API-Operation, um den `global_db_rpo` Parameter zu ändern.

Anzeigen des Recovery Point Objective (Zeitraums zwischen zwei Datensicherungen)

Das Recovery Point Objective (RPO) eines globalen Clusters wird im `global_db_rpo` Parameter für jeden DB-Cluster gespeichert.

Sie können die CLI verwenden, um den `global_db_rpo` Parameter für einen Amazon DocumentDB-Cluster anzuzeigen. Verwenden Sie die `--query` Option, um nur den `global_db_rpo` Parameter aus der Parametergruppe zurückzugeben.

Für Linux, macOS oder Unix:

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name my_custom_global_parameter_group \
```

```
    --query "Parameters[?ParameterName=='global_db_rpo']"
```

Für Windows:

```
aws docdb describe-db-cluster-parameters ^
    --db-cluster-parameter-group-name my_custom_global_parameter_group ^
    --query "Parameters[?ParameterName=='global_db_rpo']"
```

Daraufhin erhalten Sie ein Ergebnis, das dem hier dargestellten entspricht.

```
[
    {
        "ParameterName": "global_db_rpo",
        "Description": "(s) Recovery point objective threshold, in seconds, that blocks
 user commits when it is violated.",
        "Source": "engine-default",
        "ApplyType": "dynamic",
        "DataType": "integer",
        "AllowedValues": "20-2147483647",
        "IsModifiable": true,
        "ApplyMethod": "immediate"
    }
]
```

Weitere Hinweise zum Anzeigen von Parametern der Cluster-Parametergruppe finden Sie unterVerwaltung von Amazon DocumentDB-Cluster-Parametergruppen.

Deaktivieren des Recovery Point Objective

Um den RPO zu deaktivieren, setzen Sie den `global_db_rpo`-Parameter zurück. Sie können Parameter mithilfe der AWS-Managementkonsole, der oder der AWS CLI Amazon DocumentDB-API zurücksetzen.

Using the AWS-Managementkonsole

So deaktivieren Sie den RPO:

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb

2. Wählen Sie im Navigationsbereich Parameter groups (Parametergruppen) aus.
3. Wählen Sie in der Liste Ihre primäre DB-Cluster-Parametergruppe aus.
4. Wählen Sie das Optionsfeld neben dem Parameter global_db_rpo.
5. Wählen Sie Auf Standard zurücksetzen und bestätigen Sie dies.

Weitere Hinweise zum Zurücksetzen eines Parameters mit der Konsole finden Sie unterÄndern der Amazon DocumentDB-Cluster-Parametergruppen.

Using the AWS CLI

Um den Parameter `global_db_rpo` zurückzusetzen, verwenden Sie den Befehl reset-db-cluster-parameter-group.

Für Linux, macOS oder Unix:

```
aws docdb reset-db-cluster-parameter-group \
    --db-cluster-parameter-group-name my_custom_global_parameter_group \
    --parameters "ParameterName=global_db_rpo,ApplyMethod=immediate"
```

Für Windows:

```
aws docdb reset-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name my_custom_global_parameter_group ^
    --parameters "ParameterName=global_db_rpo,ApplyMethod=immediate"
```

Using the Amazon DocumentDB API

Verwenden Sie die ResetDBClusterParameterGroup API-Operation, um den `global_db_rpo` Parameter zurückzusetzen.

Verwaltung von Amazon DocumentDB-Clustern

Um einen Amazon DocumentDB-Cluster zu verwalten, benötigen Sie eine IAM-Richtlinie mit den entsprechenden Amazon DocumentDB-Steuerplane-Berechtigungen. Diese Berechtigungen ermöglichen das Erstellen, Ändern und Löschen von Clustern und Instances. Die `AmazonDocDBFullAccess` Richtlinie bietet alle erforderlichen Berechtigungen für die Verwaltung eines Amazon DocumentDB-Clusters.

Die folgenden Themen zeigen, wie Sie bei der Arbeit mit Amazon DocumentDB-Clustern verschiedene Aufgaben ausführen, darunter das Erstellen, Löschen, Ändern, Herstellen einer Verbindung zu und Anzeigen von Clustern.

Themen

- Erläuterungen zu Clustern
- Amazon DocumentDB-Cluster-Einstellungen
- Amazon DocumentDB-Cluster-Speicherkonfigurationen
- Den Status eines Clusters ermitteln
- Lebenszyklus des Amazon DocumentDB-Clusters
- Skalieren von Amazon DocumentDB-Clustern
- Klonen eines Volumes für einen Amazon DocumentDB-Cluster
- Grundlegendes zur Amazon DocumentDB-Cluster-Fehlertoleranz

Erläuterungen zu Clustern

Amazon DocumentDB trennt Rechenleistung und Speicherung und verlagert Datenreplikation und Sicherung auf das Cluster-Volume. Ein Cluster-Volume bietet eine dauerhafte, zuverlässige und hochverfügbare Speicherschicht, die Daten auf sechs Arten über drei Availability Zones repliziert. Replikate ermöglichen eine höhere Datenverfügbarkeit und Leseskalierung. Jeder Cluster kann auf bis zu 15 Replikate skaliert werden.

Substantiv	Description	API-Operationen (Verben)

Cluster	Besteht aus einer oder mehreren Instances und einem Cluster-Storage-Volume, das die Daten für diese Instances verwaltet.	`create-db-cluster`
		`delete-db-cluster`
		`describe-db-clusters`
		`modify-db-cluster`

Instance	Das Lesen und Schreiben von Daten auf das Cluster-Speichervolumen erfolgt über Instances. In einem	`create-db-instance` `delete-db-instance`

Substantiv	Description	API-Operationen (Verben)

	gegebenen Cluster gibt es zwei Arten von Instances : Primär und Replikat. Ein Cluster hat immer eine primäre Instanz und kann 0—15 Replikate haben.	`describe-db-instances`
		`modify-db-instance`
		`describe-orderable-db-instance-options`
		`reboot-db-instance`

Cluster-Volumen	Ein virtuelles Datenbank speichervolumen, das sich über drei Availability Zones erstreckt, wobei jede Availability Zone zwei Kopien der Cluster-Daten aufweist.	N/A
Primäre Instance	Unterstützt Lese- und Schreiboperationen und führt alle Datenänderungen im Cluster-Volume durch. Jeder Cluster verfügt über eine primäre Instance.	N/A
Replikat-Instance	Unterstützt nur Lesevorgänge. Jeder Amazon DocumentDB-Cluster kann zusätzlich zur primären Instance bis zu 15 Replica-Instances haben. Mehrere Replikate verteilen den Lese-Workload. Durch die Platzierung von Replikaten in separaten Availability Zones können Sie außerdem die Verfügbarkeit der Datenbank erhöhen.	N/A

Substantiv	Description	API-Operationen (Verben)
Cluster-Endpunkt	Ein Endpunkt für einen Amazon DocumentDB-Cluster , der eine Verbindung zur aktuellen primären Instance für den Cluster herstellt. Jeder Amazon DocumentDB-Cluster hat einen Cluster-Endpunkt und eine primäre Instance.	N/A
Leser-Endpunkt	Ein Endpunkt für einen Amazon DocumentDB-Cluster, der eine Verbindung zu einem der verfügbaren Replikate für diesen Cluster herstellt. Jeder Amazon DocumentD B-Cluster hat einen Reader-Endpunkt. Wenn es mehr als ein Replikat gibt, leitet der Reader-Endpunkt jede Verbindungsanforderung an eines der Amazon DocumentD B-Repliken weiter.	N/A
Instance-Endpunkt	Ein Endpunkt für eine Instance in einem Amazon DocumentD B-Cluster, die eine Verbindung zu einer bestimmten Instance herstellt. Jede Instance in einem Cluster hat unabhängi g vom Instance-Typ einen eigenen eindeutigen Instance-Endpunkt.	N/A

Amazon DocumentDB-Cluster-Einstellungen

Beim Erstellen oder Ändern eines Clusters müssen Sie wissen, welche Parameter unveränderlich und welche nach dem Anlegen des Clusters veränderbar sind. Die folgende Tabelle listet alle Einstellungen bzw. Parameter auf, die für einen Cluster spezifisch sind. Wie in der Tabelle angegeben, sind nur einige Parameter änderbar.

> **Note**
>
> Diese Einstellungen sollten nicht mit Amazon DocumentDB-Cluster-Parametergruppen und ihren Parametern verwechselt werden. Weitere Informationen zu Cluster-Parametergruppen finden Sie unter Verwaltung von Amazon DocumentDB-Cluster-Parametergruppen.

Parameter	Anpassbar	Hinweise
`DBClusterIdentifier`	Ja	Benennungseinschränkungen: • Die Länge besteht aus [1—63] Buchstaben, Zahlen oder Bindestrichen. • Muss mit einem Buchstaben beginnen. • Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten. • Muss für alle Cluster in Amazon RDS, Amazon Neptune und Amazon DocumentDB pro Region eindeutig sein. AWS-Konto
`Engine`	Nein	Der Wert muss `docdb` sein.
`BackupRetentionPeriod`	Ja	Muss zwischen 1 und 35 Tagen liegen.
`DBClusterParameterGroupName`	Ja	Benennungseinschränkungen: • Die Länge beträgt [1—255] alphanumerische Zeichen. • Muss mit einem Buchstaben beginnen.

Parameter	Anpassbar	Hinweise
		• Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.
DBSubnetGroupName	Nein	Sie können das Subnetz des Clusters nach dem Anlegen eines Clusters nicht mehr ändern.
EngineVersion	Nein	Der Wert kann `5.0.0` (Standard), `4.0.0` oder sein. `3.6.0`
KmsKeyId	Nein	Wenn Sie Ihren Cluster verschlüsseln möchten, können Sie den AWS KMS Schlüssel, den Sie zum Verschlüsseln Ihres Clusters verwendet haben, nicht ändern.
MasterUsername	Nein	Sie können den `MasterUsername` nach dem Anlegen eines Clusters nicht mehr ändern. Benennungseinschränkungen: • Die Länge beträgt [1—63] alphanumerische Zeichen. • Muss mit einem Buchstaben beginnen. • Darf kein Wort sein, das von der Datenbank-Engine reserviert ist.

MasterUserPassword	Ja	Einschränkungen:
		
		• Die Länge beträgt [8—100] druckbare ASCII-Zeichen.
		• Es können alle druckbaren ASCII-Zeichen mit Ausnahme der folgenden verwendet werden:
		  • **/** (Schrägstrich)
		  • **"** (doppeltes Anführungszeichen)
		  • **@** ('At'-Symbol)

Parameter	Anpassbar	Hinweise
`Port`	Ja	Die Portnummer gilt für alle Instances im Cluster.
`PreferredBackupWindow`	Ja	
`PreferredMaintenanceWindow`	Ja	
`StorageEncrypted`	Nein	Wenn Sie Ihren Cluster verschlüsseln, kann er nicht unverschlüsselt sein.
`StorageType`	Ja	Der Speichertyp für den DB-Cluster: Standard (`standard`) oder (). I/O-Optimized `iopt1` Standard: `standard` Dieser Parameter kann mit `CreateDBCluster` und konfiguriert werden`ModifyDBCluster` . Weitere Informationen finden Sie unter Amazon DocumentDB-Cluster-Speicherkonfigurationen.
`Tags`	Ja	
`VpcSecurityGroupIds`	Nein	Nachdem ein Cluster erstellt wurde, können Sie die VPC nicht ändern, in der sich der Cluster befindet.

Amazon DocumentDB-Cluster-Speicherkonfigurationen

Ab Amazon DocumentDB 5.0 unterstützen instanzbasierte Cluster zwei Speicherkonfigurationstypen:

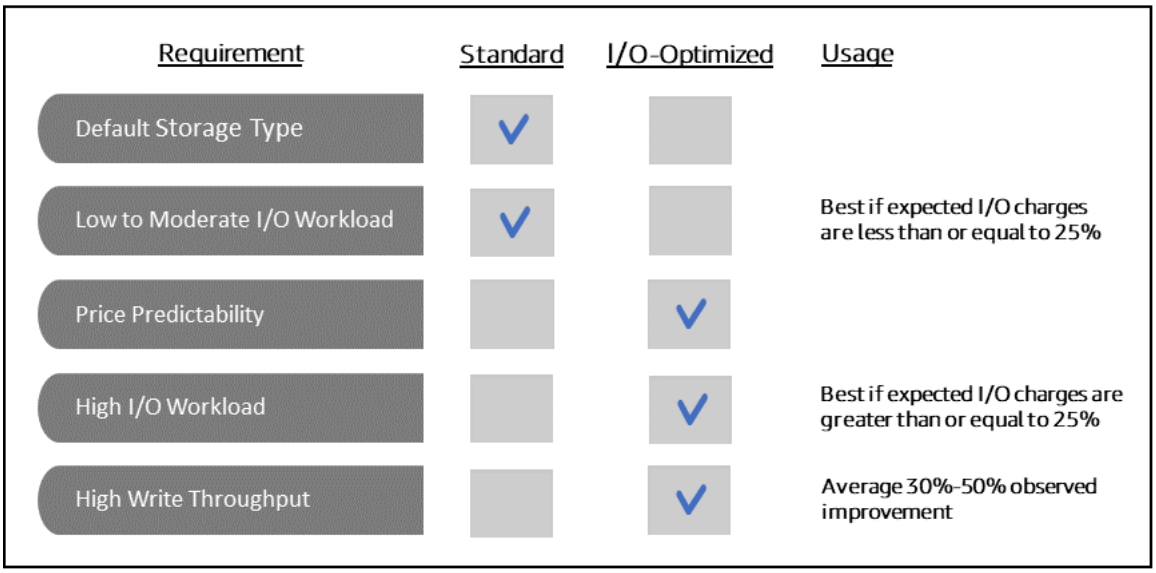
- Amazon DocumentDB-Standardspeicher: Konzipiert für Kunden mit geringem bis moderatem Verbrauch. I/O Wenn Sie davon ausgehen, dass Ihre I/O Kosten weniger als 25% Ihres gesamten Amazon DocumentDB-Clusters ausmachen, ist diese Wahl möglicherweise ideal für Sie. Bei der

Amazon DocumentDB-Standardspeicherkonfiguration wird Ihnen zusätzlich zu den Instance- und Speichergebühren eine I/O Pay-per-Request-Basis in Rechnung gestellt. Das bedeutet, dass Ihre Abrechnung je nach Nutzung von Zyklus zu Zyklus variieren kann. Die Konfiguration ist auf die schwankenden I/O Anforderungen Ihrer Anwendung zugeschnitten.

- Amazon I/O-optimized DocumentDB-Speicher: Konzipiert für Kunden, die Wert auf Preisvorhersehbarkeit legen oder intensive Anwendungen haben. I/O Die I/O-optimized Konfiguration bietet eine verbesserte Leistung, einen höheren Durchsatz und eine geringere Latenz für Kunden mit I/O intensiven Workloads. Wenn Sie davon ausgehen, dass Ihre I/O Kosten 25% Ihrer gesamten Amazon DocumentDB-Cluster-Kosten übersteigen, bietet diese Option ein besseres Preis-Leistungs-Verhältnis. Mit der Amazon I/O-optimized DocumentDB-Speicherkonfiguration werden Ihnen keine I/O Betriebskosten in Rechnung gestellt, sodass die Kosten in jedem Abrechnungszeitraum vorhersehbar sind. Die Konfiguration stabilisiert die Kosten und verbessert gleichzeitig die Leistung.

Sie können Ihre vorhandenen Datenbank-Cluster einmal alle 30 Tage auf Amazon I/O-optimized DocumentDB-Speicher umstellen. Sie können jederzeit wieder zum Amazon DocumentDB-Standardspeicher wechseln. Das nächste Datum, an dem die Speicherkonfiguration geändert werden muss, I/O-optimized kann mit dem `describe-db-clusters` Befehl auf AWS CLI oder über die auf der AWS-Managementkonsole Konfigurationsseite des Clusters nachverfolgt werden.

Sie können ein neues Datenbank-Cluster einschließlich der Amazon I/O-optimized DocumentDB-Konfiguration erstellen oder Ihre vorhandenen Datenbank-Cluster mit wenigen Klicks AWS-Managementkonsole, einer einzigen Parameteränderung in der AWS Command Line Interface (AWS CLI) oder über AWS SDKs konvertieren. Während oder nach der Änderung der Speicherkonfiguration sind keine Ausfallzeiten oder ein Neustart der Instances erforderlich.

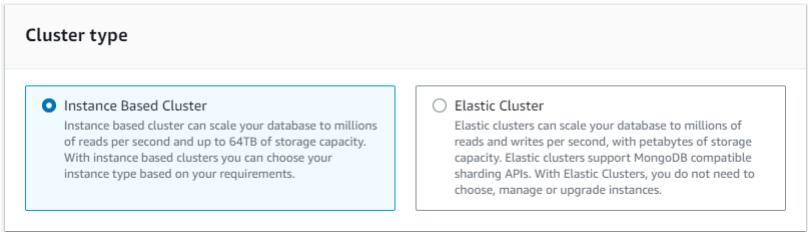


Einen Cluster erstellen I/O-optimized

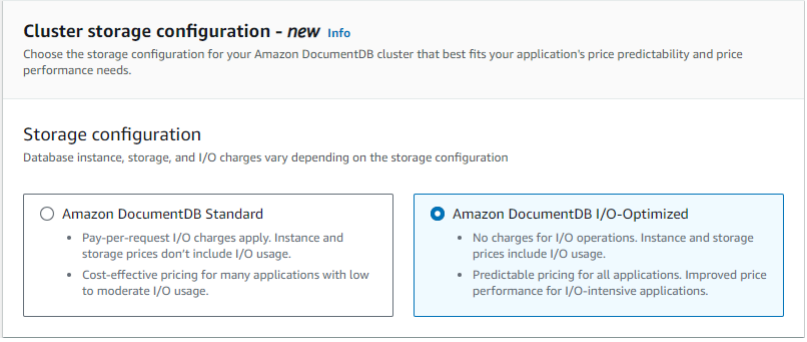
Using the AWS-Managementkonsole

Um einen I/O-optimized Cluster zu erstellen oder zu ändern, verwenden Sie AWS-Managementkonsole:

1. Wählen Sie in der Amazon DocumentDB-Managementkonsole unter Clusters entweder Create oder wählen Sie den Cluster aus und wählen Sie Actions und dann Modify.
2. Wenn Sie einen neuen Cluster erstellen, stellen Sie sicher, dass Sie im Abschnitt Clustertyp die Option Instance Based Cluster auswählen (dies ist die Standardoption).



3. Wählen Sie im Abschnitt Konfiguration unter Cluster-Speicherkonfiguration die Option Amazon DocumentDB I/O-Optimized aus.



4. Schließen Sie die Clustererstellung oder -änderung ab und wählen Sie Cluster erstellen oder Cluster ändern.

   Den vollständigen Vorgang „Cluster erstellen“ finden Sie unterErstellen eines Clusters und einer primären Instance mit dem AWS-Managementkonsole.

   Den vollständigen Vorgang „Cluster modifizieren“ finden Sie unterÄndern eines Amazon DocumentDB-Clusters.

Using the AWS CLI

So erstellen Sie einen I/O-optimized Cluster mit dem AWS CLI:

Ersetzen Sie in den folgenden Beispielen jedes *user input placeholder* durch Ihre eigenen Informationen.

Für Linux, macOS oder Unix:

```
aws docdb create-db-cluster \
      --db-cluster-identifier sample-cluster \
      --engine docdb \
      --engine-version 5.0.0 \
      --storage-type iopt1 \
      --deletion-protection \
      --master-username username \
      --master-user-password password
```

Für Windows:

```
aws docdb create-db-cluster ^
      --db-cluster-identifier sample-cluster ^
      --engine docdb ^
      --engine-version 5.0.0 ^
```

```
    --storage-type iopt1 ^
    --deletion-protection ^
    --master-username username ^
    --master-user-password password
```

Kostenanalyse zur Bestimmung der Speicherkonfiguration

Mit Amazon DocumentDB haben Sie die Flexibilität, Ihre Speicherkonfiguration für jeden Datenbank-Cluster auszuwählen, den Sie haben. Um Ihre Cluster richtig zwischen Standard und zuzuordnen I/O-optimized, können Sie Ihre Amazon DocumentDB-Kosten clusterweise verfolgen. Dazu können Sie Tags zu vorhandenen Clustern hinzufügen, das Tagging für die Kostenzuweisung in Ihrem AWS Fakturierung und Kostenmanagement Dashboard aktivieren und Ihre Kosten für einen bestimmten Cluster im analysieren. AWS Cost Explorer Service Informationen zur Kostenanalyse finden Sie in unserem Blog Verwenden von Kostenzuweisungs-Tags.

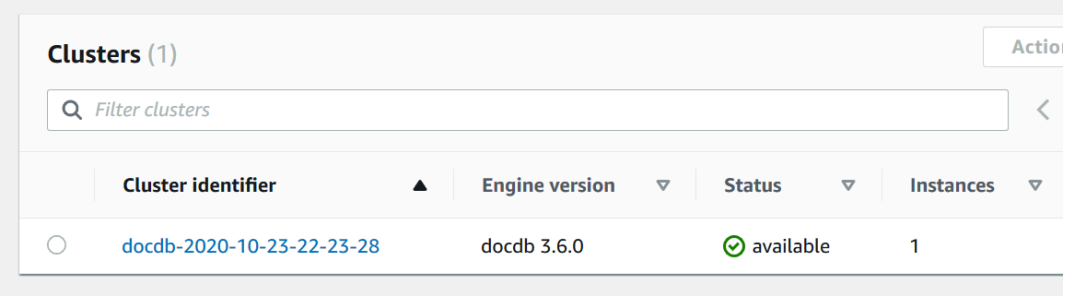
Den Status eines Clusters ermitteln

Sie können den Status eines Clusters mithilfe von AWS-Managementkonsole oder ermitteln AWS CLI.

Using the AWS-Managementkonsole

Verwenden Sie das folgende Verfahren, um den Status Ihres Amazon DocumentDB-Clusters mithilfe des AWS-Managementkonsole

1. Melden Sie sich bei und öffnen Sie die Amazon DocumentDB-Konsole unter. AWS-Managementkonsole https://console.aws.amazon.com/docdb
2. Klicken Sie im Navigationsbereich auf Cluster.
3. Suchen Sie in der Spalte Cluster identifier (Cluster-ID) den Namen des Clusters, der Sie interessiert. Um dann den Status des Clusters zu ermitteln, lesen Sie diese Zeile in die Spalte Status, wie unten gezeigt.



Using the AWS CLI

Verwenden Sie den `describe-db-clusters` Vorgang, um den Status Ihres Amazon DocumentDB-Clusters mithilfe von zu überprüfen. AWS CLI

Der folgende Code ermittelt den Status des Clusters `sample-cluster`.

Für Linux, macOS oder Unix:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster  \
    --query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

Für Windows:

```
aws docdb describe-db-clusters ^
    --db-cluster-identifier sample-cluster  ^
    --query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
[
    [
        "sample-cluster",
        "available"
    ]
]
```

Lebenszyklus des Amazon DocumentDB-Clusters

Der Lebenszyklus eines Amazon DocumentDB-Clusters umfasst das Erstellen, Beschreiben, Ändern und Löschen des Clusters. Dieser Abschnitt enthält Informationen darüber, wie Sie diese Prozesse abschließen können.

Themen

- Erstellen eines Amazon DocumentDB-Clusters
- Beschreibung von Amazon DocumentDB-Clustern
- Ändern eines Amazon DocumentDB-Clusters
- Feststellung ausstehender Wartungsarbeiten

- Durchführung eines Patch-Updates für die Engine-Version eines Clusters
- Das Betriebssystem eines Clusters wird aktualisiert
- Stoppen und Starten eines Amazon DocumentDB-Clusters
- Löschen eines Amazon DocumentDB-Clusters

Erstellen eines Amazon DocumentDB-Clusters

Ein Amazon DocumentDB-Cluster besteht aus Instances und einem Cluster-Volume, das die Daten für den Cluster darstellt. Das Cluster-Volumen wird auf sechs Arten über drei Availability Zones als ein einziges, virtuelles Volume repliziert. Der Cluster enthält eine primäre Instance und optional bis zu 15 Replikat-Instances.

In den folgenden Abschnitten wird gezeigt, wie Sie einen Amazon DocumentDB-Cluster mithilfe von AWS-Managementkonsole oder erstellen. AWS CLI Sie können dann weitere Replikat-Instances für diesen Cluster hinzufügen. Wenn Sie die Konsole verwenden, um Ihren Amazon DocumentDB-Cluster zu erstellen, wird gleichzeitig automatisch eine primäre Instance für Sie erstellt. Wenn Sie das verwenden, AWS CLI um Ihren Amazon DocumentDB-Cluster zu erstellen, müssen Sie, nachdem der Status des Clusters verfügbar ist, die primäre Instance für diesen Cluster erstellen.

Voraussetzungen

Im Folgenden sind Voraussetzungen für die Erstellung eines Amazon DocumentDB-Clusters aufgeführt.

Wenn Sie keinen haben AWS-Konto, führen Sie die folgenden Schritte aus, um einen zu erstellen.

Um sich für eine anzumelden AWS-Konto

1. Offen https://signin.aws.amazon.com/signup? request_type=registrieren.
2. Folgen Sie den Online-Anweisungen.

   Während der Anmeldung erhalten Sie einen Telefonanruf oder eine Textnachricht und müssen einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für eine registrieren AWS-Konto, wird eine erstellt. Root-Benutzer des AWS-Kontos Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Benutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

VPC-Voraussetzungen

Sie können einen Amazon DocumentDB-Cluster nur in einer Amazon Virtual Private Cloud (Amazon VPC) erstellen. Ihre Amazon VPC muss über mindestens ein Subnetz in jeder der mindestens zwei Availability Zones verfügen, damit Sie sie mit einem Amazon DocumentDB-Cluster verwenden können. Indem Sie Ihre Cluster-Instances auf Availability Zones verteilen, stellen Sie sicher, dass Instances im unwahrscheinlichen Fall eines Ausfalls der Availability Zone in Ihrem Cluster verfügbar sind.

Voraussetzungen für das Subnetz

Wenn Sie einen Amazon DocumentDB-Cluster erstellen, müssen Sie eine VPC und die entsprechende Subnetzgruppe innerhalb dieser VPC auswählen, um Ihren Cluster zu starten. Subnetze bestimmen die Availability Zone und den IP-Bereich innerhalb dieser Availability Zone, die Sie zum Starten einer Instance verwenden möchten. Im Rahmen dieser Ausführungen werden die Begriffe Subnetz und Availability Zone synonym verwendet. Eine Subnetzgruppe ist ein benannter Satz von Subnetzen (oder Availability Zones). Mit einer Subnetzgruppe können Sie lediglich die Availability Zones angeben, die Sie für den Start von Amazon DocumentDB-Instances verwenden möchten. So wird beispielsweise in einem Cluster mit drei Instances für eine hohe Verfügbarkeit empfohlen, dass jede dieser Instances in separaten Availability Zones bereitgestellt wird. Wenn eine einzelne Availability Zones ausfällt, wirkt sich dies nur auf eine einzelne Instance aus.

Amazon DocumentDB-Instances können derzeit in bis zu drei Availability Zones bereitgestellt werden. Selbst wenn eine Subnetzgruppe mehr als drei Subnetze hat, können Sie nur drei dieser Subnetze verwenden, um einen Amazon DocumentDB-Cluster zu erstellen. Daher wird empfohlen, dass Sie beim Anlegen einer Subnetzgruppe nur die drei Subnetze auswählen, in denen Sie Ihre Instances bereitstellen möchten. In den USA Ost (Nord-Virginia) kann Ihre Subnetzgruppe sechs Subnetze (oder Availability Zones) haben. Wenn jedoch ein Amazon DocumentDB-Cluster bereitgestellt wird, wählt Amazon DocumentDB drei dieser Availability Zones aus, die es zur Bereitstellung von Instances verwendet.

Nehmen wir beispielsweise an, dass Amazon DocumentDB beim Erstellen eines Clusters die Availability Zones {1A, 1B und 1C} auswählt. Wenn Sie nun versuchen, eine Instance in Availability Zone {1D} zu erstellen, schlägt der API-Aufruf fehl. Wenn Sie sich jedoch dafür entscheiden, eine Instance zu erstellen, ohne eine bestimmte Availability Zone anzugeben, wählt Amazon DocumentDB in Ihrem Namen eine Availability Zone aus. Amazon DocumentDB verwendet einen Algorithmus für den Lastenausgleich der Instances über die Availability Zones hinweg, um Ihnen zu helfen, eine hohe Verfügbarkeit zu erreichen. Wenn beispielsweise drei Instances bereitgestellt werden, werden diese

standardmäßig über drei Availability Zones verteilt und nicht alle in einer einzigen Availability Zone bereitgestellt.

Empfehlungen:

- Wenn Sie keinen speziellen Grund haben, legen Sie immer eine Subnetzgruppe mit drei Subnetzen an. Auf diese Weise wird sichergestellt, dass Cluster mit drei oder mehr Instances eine höhere Verfügbarkeit erreichen können, da Instances über drei Availability Zones bereitgestellt werden.
- Verteilen Sie Instances immer über mehrere Availability Zones, um eine hohe Verfügbarkeit zu erreichen. Platzieren Sie niemals alle Instances für einen Cluster in einer einzigen Availability Zone.
- Da Failover-Ereignisse jederzeit auftreten können, sollten Sie nicht davon ausgehen, dass sich eine primäre Instance oder Replikat-Instance immer in einer bestimmten Availability Zone befinden.

Zusätzliche Voraussetzungen

Im Folgenden sind einige zusätzliche Voraussetzungen für die Erstellung eines Amazon DocumentDB-Clusters aufgeführt:

- Wenn Sie AWS mithilfe von AWS Identity and Access Management (IAM-) Anmeldeinformationen eine Verbindung herstellen, muss Ihr IAM-Konto über IAM-Richtlinien verfügen, die die für die Ausführung von Amazon DocumentDB-Vorgängen erforderlichen Berechtigungen gewähren.

  Wenn Sie ein IAM-Konto für den Zugriff auf die Amazon DocumentDB-Konsole verwenden, müssen Sie sich zuerst mit Ihrem IAM-Konto bei der anmelden. AWS-Managementkonsole Gehen Sie dann zur Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
- Wenn Sie die Konfigurationsparameter für Ihren Cluster anpassen möchten, müssen Sie eine Cluster-Parametergruppe und eine Parametergruppe mit den erforderlichen Parametereinstellungen festlegen. Informationen zum Erstellen oder Ändern einer Cluster-Parametergruppe oder Parametergruppe finden Sie unter Verwaltung von Amazon DocumentDB-Cluster-Parametergruppen.
- Sie müssen die TCP/IP Portnummer ermitteln, die Sie für Ihren Cluster angeben möchten. Die Firewalls einiger Unternehmen blockieren Verbindungen zu den Standard-Ports für Amazon DocumentDB. Wenn die Firewall Ihres Unternehmens den Standard-Port blockiert, wählen Sie einen anderen Port für Ihr Cluster aus. Alle Instances in einem Cluster verwenden denselben Port.

Erstellen eines Clusters und einer primären Instance mit dem AWS-Managementkonsole

Die folgenden Verfahren beschreiben, wie Sie die Konsole verwenden, um einen Amazon DocumentDB-Cluster mit einer oder mehreren Instances zu starten.

Erstellen Sie einen Cluster: Verwenden Sie die Standardeinstellungen

Um einen Cluster mit Instanzen zu erstellen, die die Standardeinstellungen verwenden, verwenden Sie den AWS-Managementkonsole

1. Melden Sie sich bei und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb. AWS-Managementkonsole
2. Wenn Sie Ihren Cluster in einer AWS-Region anderen Region als der Region USA Ost (Nord-Virginia) erstellen möchten, wählen Sie die Region aus der Liste im oberen rechten Bereich der Konsole aus.
3. Wählen Sie im Navigationsbereich Clusters (Cluster) und dann Create (Erstellen).

   > **Tip**
   >
   > Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

4. Füllen Sie auf der Seite „Amazon DocumentDB-Cluster erstellen“ den Bereich „Konfiguration“ aus.

   a. Cluster-ID — Akzeptieren Sie den von Amazon DocumentDB bereitgestellten Namen, oder geben Sie einen Namen für Ihren Cluster ein, z. B. **`sample-cluster`**

      Einschränkungen bei der Benennung von Clustern:

      - Die Länge besteht aus [1—63] Buchstaben, Zahlen oder Bindestrichen.
      - Muss mit einem Buchstaben beginnen.
      - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.
      - Muss für alle Cluster in Amazon RDS, Neptune und Amazon DocumentDB pro Region eindeutig sein. AWS-Konto

b. Engine-Version — Akzeptieren Sie die Standard-Engine-Version 5.0.0 oder wählen Sie optional 8.0.0, 4.0.0 oder 3.6.0.

c. Instanzklasse — Übernehmen Sie die Standardeinstellung`db.r5.large`, oder wählen Sie die gewünschte Instanzklasse aus der Liste aus.

d. Anzahl der Instanzen — Wählen Sie in der Liste die Anzahl der Instanzen aus, die Sie mit diesem Cluster erstellen möchten. Die erste Instance ist die primäre Instance. Alle anderen Instances sind schreibgeschützte Replikat-Instances. Sie können später Instances hinzufügen und löschen. Standardmäßig wird ein Amazon DocumentDB-Cluster mit drei Instances (einer primären und zwei Repliken) gestartet.

5. Füllen Sie den Abschnitt zur Cluster-Speicherkonfiguration aus.

   Wählen Sie entweder Amazon DocumentDB Standard (Standard) oder Amazon I/O-Optimized DocumentDB. Weitere Informationen finden Sie unter Amazon DocumentDB-Cluster-Speicherkonfigurationen.

6. Vervollständigen Sie den Bereich Authentication (Authentifizierung).

   a. Nutzername — Geben Sie einen Namen für den Hauptbenutzer ein. Um sich bei Ihrem Cluster anzumelden, müssen Sie den primären Benutzernamen verwenden.

      Einschränkungen bei der Benennung primärer Benutzer:

      - Die Länge beträgt [1—63] alphanumerische Zeichen.
      - Muss mit einem Buchstaben beginnen.
      - Darf kein Wort sein, das von der Datenbank-Engine reserviert ist.

   b. Wählen Sie eine der folgenden Passwortoptionen:

      - Verwaltet in AWS Secrets Manager — Wählen Sie diese Option, wenn Sie Ihr Hauptbenutzerkennwort automatisch verwalten möchten AWS Secrets Manager .

        Wenn Sie diese Option wählen, konfigurieren Sie den KMS-Schlüssel, indem Sie entweder Ihren eigenen erstellen oder einen Schlüssel verwenden, den Secrets Manager erstellt.

      - Selbstverwaltet — Wählen Sie diese Option, wenn Sie Ihr primäres Benutzerkennwort selbst verwalten möchten. Wenn Sie diese Option wählen, geben Sie ein Passwort für den Hauptbenutzer ein und bestätigen Sie es dann. Um sich bei Ihrem Cluster anzumelden, müssen Sie das Passwort für den Hauptbenutzer verwenden.

Passwortbeschränkungen:

- Länge beträgt [8–100] druckbare ASCII-Zeichen.
- Es können alle druckbaren ASCII-Zeichen mit Ausnahme der folgenden verwendet werden:
  - **/** (Schrägstrich)
  - **"** (doppeltes Anführungszeichen)
  - **@** ('At'-Symbol)

7. Wählen Sie am unteren Rand der Seite eine der folgenden Optionen:

- Um den Cluster jetzt zu erstellen, wählen Sie Create Cluster (Cluster erstellen) aus.
- Wenn der Cluster nicht neu erstellt werden soll, wählen Sie Cancel (Abbrechen) aus.
- Um den Cluster vor der Erstellung weiter zu konfigurieren, wählen Sie Show additional configurations (Weitere Konfigurationen anzeigen) aus und fahren dann mit Erstellen Sie einen Cluster: zusätzliche Konfigurationen fort.

  Im Bereich Additional Configurations (Zusätzliche Konfigurationen) sind die folgenden Konfigurationen zu finden:

  - Netzwerkeinstellungen — Standardmäßig wird die `default` VPC-Sicherheitsgruppe verwendet.
  - Clusteroptionen — Standardmäßig wird der Port 27017 und die Standardparametergruppe verwendet.
  - Verschlüsselung — In der Standardeinstellung wird die Verschlüsselung mit dem `(default) aws/rds` Schlüssel aktiviert.

    > **Important**
    >
    > Die Verschlüsselung eines einmal verschlüsselten Clusters kann nicht wieder aufgehoben werden.

  - Backup — Standardmäßig werden Backups 1 Tag lang aufbewahrt und Amazon DocumentDB kann das Backup-Fenster wählen.
  - Protokollexporte — Standardmäßig werden keine Audit-Logs in Logs exportiert CloudWatch .
  - Wartung — Standardmäßig wählt Amazon DocumentDB das Wartungsfenster.

- Löschschutz — Schützen Sie Ihren Cluster vor versehentlichem Löschen. Cluster, die mit der Konsole erstellt wurden, sind standardmäßig aktiviert.

Wenn Sie die Standardeinstellungen jetzt übernehmen, können Sie die meisten davon später durch Bearbeitung des Clusters wieder ändern.

8. Aktivieren Sie eingehende Verbindungen für die Sicherheitsgruppe Ihres Clusters.

Wenn Sie die Standardeinstellungen für Ihren Cluster nicht geändert haben, haben Sie einen Cluster mit der Standardsicherheitsgruppe für die Standard-VPC in der angegebenen Region erstellt. Um eine Verbindung zu Amazon DocumentDB herzustellen, müssen Sie eingehende Verbindungen auf Port 27017 (oder dem Port Ihrer Wahl) für die Sicherheitsgruppe Ihres Clusters aktivieren.

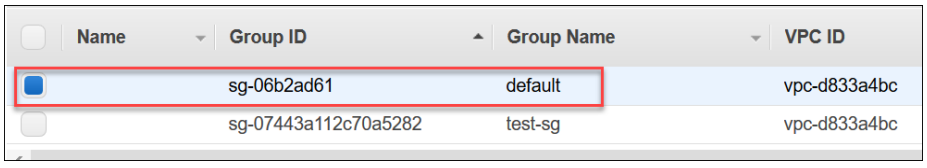
So fügen Sie eine eingehende Verbindung zur SicherheitsgruppeIhres Clusters hinzu:

a. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Amazon EC2-Konsole unter. https://console.aws.amazon.com/ec2/

b. Wählen Sie im Abschnitt Resources (Ressourcen) des Hauptfensters die Option Security groups (Sicherheitsgruppen).



c. Wählen Sie in der Liste der Sicherheitsgruppen die Sicherheitsgruppe aus, die Sie bei der Erstellung Ihres Cluster verwendet haben (höchstwahrscheinlich handelt es sich dabei um die Standard-Sicherheitsgruppe), und wählen Sie das Feld links neben dem Name der Sicherheitsgruppe aus.



d. Wählen Sie im Menü Aktionen die Option Eingangsregeln bearbeiten aus. Wählen Sie dann die Regeleinschränkungen aus oder geben Sie diese ein.

i. Typ — Wählen Sie aus der Liste das Protokoll aus, das für den Netzwerkverkehr geöffnet werden soll.

ii. Protokoll — Wählen Sie den Protokolltyp aus der Liste aus.

iii. Portbereich — Geben Sie für eine benutzerdefinierte Regel eine Portnummer oder einen Portbereich ein. Stellen Sie sicher, dass die Portnummer bzw. der Portbereich den Port enthält, den Sie beim Erstellen des Clusters angegeben haben (Standard: 27017).

iv. Quelle — Gibt den Datenverkehr an, der Ihre Instance erreichen kann. Wählen Sie die Datenverkehrquelle aus der Liste aus. Wenn Sie Benutzerdefiniert wählen, geben Sie eine einzelne IP-Adresse oder einen IP-Adressbereich in CIDR-Notation an (z. B. 203.0.113). 5/32).

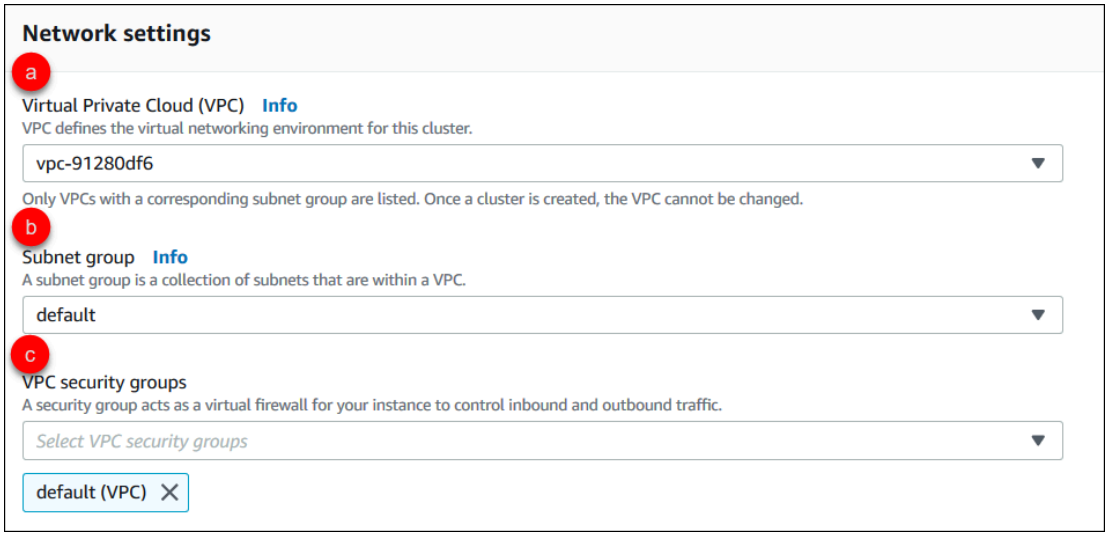
v. Beschreibung — Geben Sie eine Beschreibung für diese Regel ein.

vi. Wenn Sie die Regel fertig erstellt haben, wählen Sie Save (Speichern).

Erstellen Sie einen Cluster: zusätzliche Konfigurationen

Wenn Sie die Standardeinstellungen für Ihren Cluster übernehmen möchten, können Sie die folgenden Schritte überspringen und Create cluster (Cluster erstellen) auswählen.

1. Vervollständigen Sie den Bereich Network settings (Netzwerkeinstellungen).

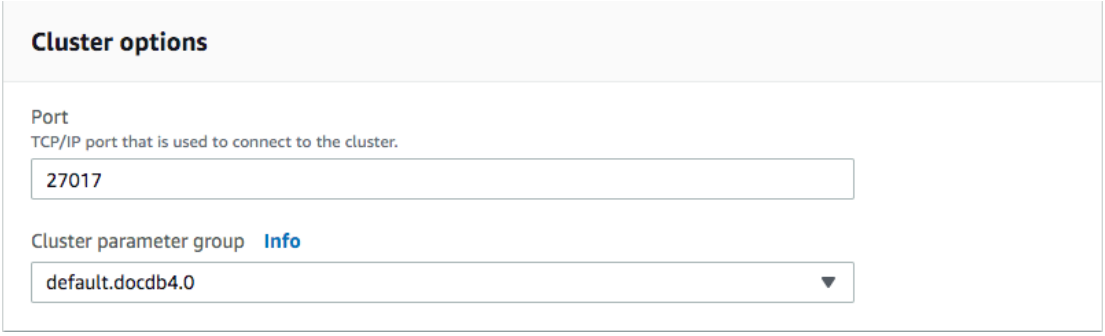


a. Virtual Private Cloud (VPC) — Wählen Sie in der Liste die Amazon VPC aus, in der Sie diesen Cluster starten möchten.

b. Subnetzgruppe — Wählen Sie in der Liste die Subnetzgruppe aus, die Sie für diesen Cluster verwenden möchten.

c. VPC-Sicherheitsgruppen — Wählen Sie in der Liste die VPC-Sicherheitsgruppe für diesen Cluster aus.

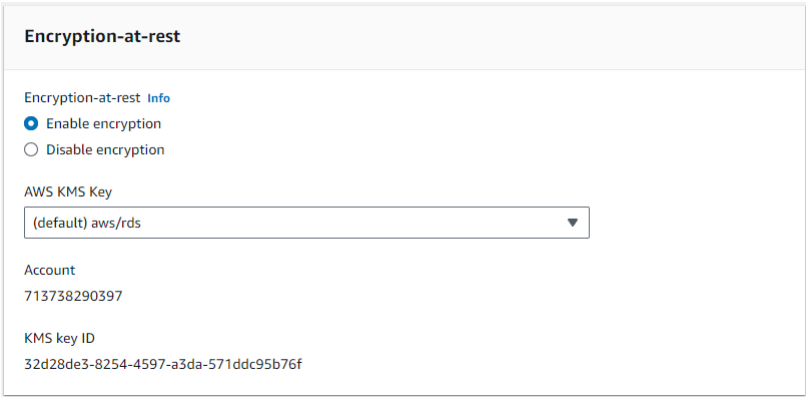
2. Vervollständigen Sie den Bereich Cluster options (Cluster-Optionen).



a. Datenbankport — Verwenden Sie die Aufwärts- und Abwärtspfeile, um den TCP/IP Port festzulegen, den Anwendungen für die Verbindung zu Ihrer Instance verwenden.

b. Cluster-Parametergruppe — Wählen Sie in der Liste der Parametergruppen die Cluster-Parametergruppe für diesen Cluster aus.

3. Vervollständigen Sie den Bereich Encryption (Verschlüsselung).



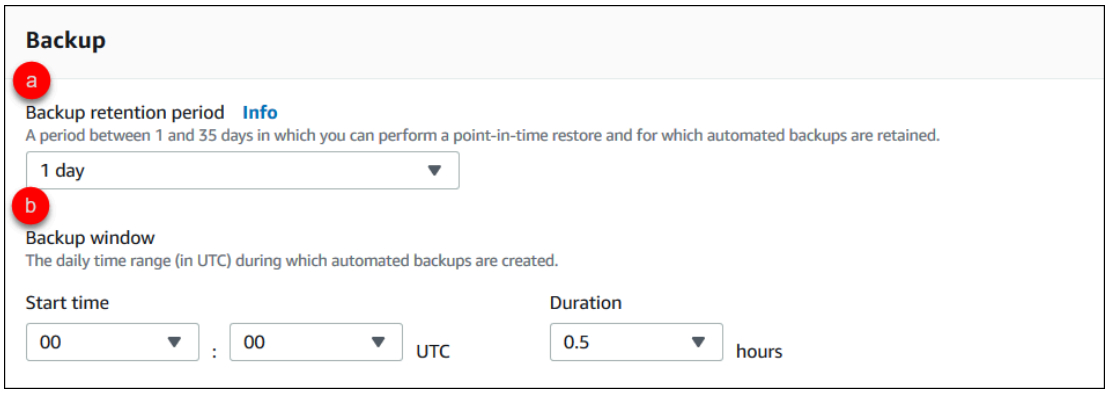
a. Encryption-at-rest— Wählen Sie eine der folgenden Optionen:

- Verschlüsselung aktivieren — Standard. Alle Daten im Ruhezustand werden verschlüsselt. Wenn Sie sich für die Verschlüsselung Ihrer Daten entscheiden, können Sie diese Aktion nicht rückgängig machen.
- Verschlüsselung deaktivieren — Ihre Daten sind nicht verschlüsselt.

b. AWS KMS-Schlüssel — Dieser ist nur verfügbar, wenn Sie Ihre Daten verschlüsseln. Wählen Sie aus der Liste den Schlüssel aus, den Sie für die Verschlüsselung der Daten in diesem Cluster verwenden möchten. Der Standardwert ist `(default) aws/rds`.

   Wenn Sie Enter a key ARN (Einen Schlüssel-ARN eingeben) auswählen, müssen Sie einen Amazon-Ressourcenname (ARN) für den Schlüssel eingeben.

4. Vervollständigen Sie den Bereich Backup.



a. Aufbewahrungszeitraum für Backups — Wählen Sie in der Liste die Anzahl der Tage aus, für die automatische Backups dieses Clusters aufbewahrt werden sollen, bevor sie gelöscht werden.

b. Backup-Fenster — Legen Sie die tägliche Uhrzeit und Dauer fest, während der Amazon DocumentDB Backups dieses Clusters erstellen soll.

   i. Startzeit — Wählen Sie in der ersten Liste die Startzeit (Stunde) (UTC) für den Start Ihrer automatischen Backups aus. Wählen Sie in der zweiten Liste die Minute für den Beginn der automatischen Backups aus.

   ii. Dauer — Wählen Sie in der Liste die Anzahl der Stunden aus, die der Erstellung automatischer Backups zugewiesen werden sollen.

5. Füllen Sie den Bereich Protokollexporte aus, indem Sie die Protokolltypen auswählen, die Sie in CloudWatch Logs exportieren möchten.



- Audit-Logs — Wählen Sie diese Option, um den Export von Audit-Logs nach Amazon CloudWatch Logs zu ermöglichen. Wenn Sie Audit logs (Auditprotokolle), auswählen, müssen Sie `audit_logs` in der benutzerdefinierten Parametergruppe des Clusters

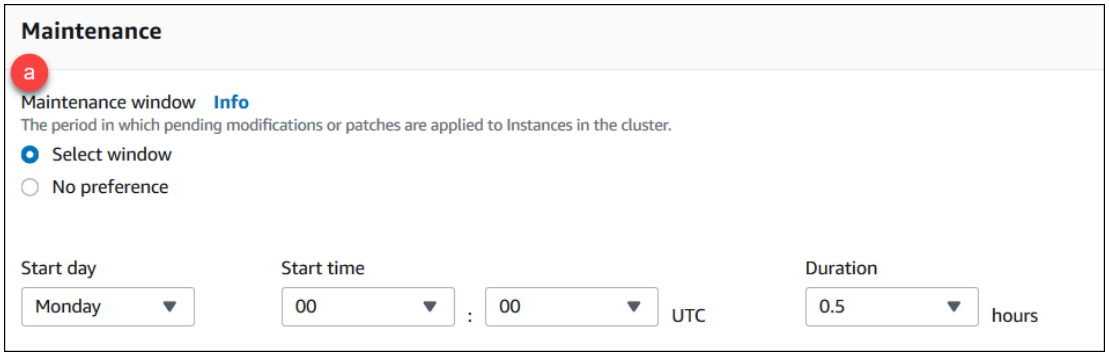
aktivieren. Weitere Informationen finden Sie unter Prüfung von Amazon DocumentDB-Ereignissen.

- Profiler-Protokolle — Wählen Sie diese Option, um den Export von Betriebsprofiler-Protokollen nach Amazon Logs zu ermöglichen. CloudWatch Wenn Sie Profiler logs (Profiler-Protokolle) auswählen, müssen Sie auch die folgenden Parameter in der benutzerdefinierten Parametergruppe des Clusters ändern:

  - `profiler`— Auf eingestellt. `enabled`
  - `profiler_threshold_ms`— Auf einen Wert setzen, `[0-INT_MAX]` um den Schwellenwert für Profilerstellungsoperationen festzulegen.
  - `profiler_sampling_rate`— Auf einen Wert setzen, um den Prozentsatz der langsamen Operationen festzulegen`[0.0-1.0]`, für die ein Profil erstellt werden soll.

  Weitere Informationen finden Sie unter Profilierung von Amazon DocumentDB-Vorgängen.

6. Schließen Sie den Bereich Maintenance (Wartung) ab.



- Wählen Sie eine der folgenden Optionen aus:

  - Fenster auswählen — Sie können den Wochentag, die UTC-Startzeit und die Dauer angeben, für die Amazon DocumentDB Wartungsarbeiten an Ihrem Cluster durchführen soll.

    a. Starttag — Wählen Sie in der Liste den Wochentag aus, an dem mit der Cluster-Wartung begonnen werden soll.

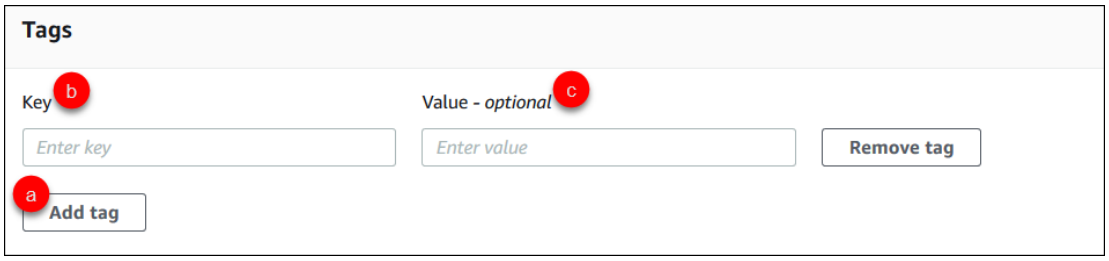
    b. Startzeit — Wählen Sie in den Listen die Stunde und Minute (UTC) aus, um mit der Wartung zu beginnen.

    c. Dauer — Wählen Sie in der Liste aus, wie viel Zeit für die Cluster-Wartung eingeplant werden soll. Wenn die Wartung nicht innerhalb der angegebenen Zeit abgeschlossen

werden kann, wird der Wartungsprozess über die angegebene Zeit hinaus bis zum Abschluss fortgesetzt.

- Keine Präferenz — Amazon DocumentDB wählt den Wochentag, die Startzeit und die Dauer der Wartungsarbeiten aus.

7. Wenn Sie diesem Cluster ein oder mehrere Tags hinzufügen möchten, stellen Sie den Bereich Tags fertig.



Wiederholen Sie die folgenden Schritte für alle Tags, die Sie dem Cluster hinzufügen möchten. Sie können bis zu 10 in einem Cluster haben.

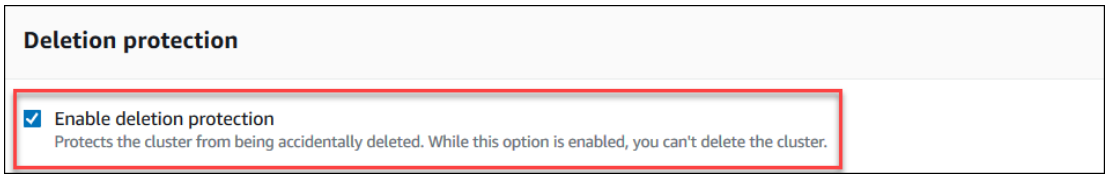
a. Wählen Sie Tags hinzufügen aus.

b. Geben Sie den Schlüssel des Tags ein.

c. Optional können Sie auch den Wert des Tags eingeben.

Klicken Sie zum Entfernen eines Tags auf Tag entfernen.

8. Der Löschschutz ist standardmäßig aktiviert, wenn Sie ein Cluster mit der Konsole erstellen. Deaktivieren Sie zum Ausschalten des Löschschutzes die Option Löschschutz aktivieren. Wenn diese Option aktiviert ist, verhindert der Löschschutz das Löschen eines Clusters. Wenn Sie einen löschgeschützten Cluster löschen möchten, müssen Sie zuerst den Cluster-Löschschutz deaktivieren.



Weitere Informationen über den Löschschutz finden Sie unter Löschen eines Amazon DocumentDB-Clusters.

9. Wählen Sie zum Erstellen des Clusters Cluster erstellen aus. Wählen Sie andernfalls Abbrechen.

Erstellen eines Clusters mit dem AWS CLI

Die folgenden Verfahren beschreiben, wie Sie den verwenden, um einen Amazon DocumentDB-Cluster AWS CLI zu starten und ein Amazon DocumentDB-Replikat zu erstellen.

Parameters

- **`--db-cluster-identifier`**—Erforderlich. Eine Zeichenfolge in Kleinbuchstaben, die diesen Cluster identifiziert.

  Einschränkungen bei der Benennung von Clustern:

  - Die Länge besteht aus [1—63] Buchstaben, Zahlen oder Bindestrichen.
  - Muss mit einem Buchstaben beginnen.
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.
  - Muss für alle Cluster (in Amazon RDS, Amazon Neptune und Amazon DocumentDB) pro Konto und Region eindeutig sein. AWS
- **`--engine`**—Erforderlich. Der Wert muss **`docdb`** sein.
- **`--deletion-protection | --no-deletion-protection`**— Fakultativ. Der aktivierte Löschschutz verhindert, dass ein Cluster gelöscht wird. Wenn Sie den verwenden AWS CLI, ist der Löschschutz in der Standardeinstellung deaktiviert.

  Weitere Informationen über den Löschschutz finden Sie unter Löschen eines Amazon DocumentDB-Clusters.
- **`--storage-type standard | iopt1`**— Fakultativ. Standard: **`standard`**. Die Speicherkonfiguration des Clusters. Gültige Werte sind `standard` (Standard) oder `iopt1` (I/O-optimized).
- **`--master-username`**—Erforderlich. Der Benutzername für die Authentifizierung des Benutzers.

  Namenseinschränkungen für Benutzer:

  - Die Länge beträgt [1 bis 63] alphanumerische Zeichen.
  - Muss mit einem Buchstaben beginnen.
  - Darf kein Wort sein, das von der Datenbank-Engine reserviert ist.
- **`--master-user-password`**— Fakultativ. Das Passwort für die Authentifizierung des Benutzers.

Einschränkungen für Hauptpasswort:

- Länge beträgt [8–100] druckbare ASCII-Zeichen.
- Es können alle druckbaren ASCII-Zeichen mit Ausnahme der folgenden verwendet werden:
  - **/** (Schrägstrich)
  - **"** (doppeltes Anführungszeichen)
  - **@** ('At'-Symbol)

- **`--manage-master-user-password`**— Fakultativ. Amazon DocumentDB generiert das Masterbenutzerkennwort und verwaltet es während seines gesamten Lebenszyklus in Secrets Manager.

Weitere Parameter finden Sie unter https://docs.aws.amazon.com/documentdb/latest/APIReference/API_CreateDBCluster.html CreateDBCluster.

Um einen Amazon DocumentDB-Cluster zu starten, verwenden Sie den AWS CLI

Um einen Amazon DocumentDB-Cluster zu erstellen, rufen Sie den auf. `create-db-cluster` AWS CLI Mit dem folgenden AWS CLI Befehl wird ein Amazon DocumentDB-Cluster `sample-cluster` mit dem Namen erstellt, wobei der Löschschutz aktiviert ist. Weitere Informationen zum Löschschutz finden Sie unterLöschen eines Amazon DocumentDB-Clusters.

Außerdem `--engine-version` ist dies ein optionaler Parameter, der standardmäßig auf die neueste Version der Hauptengine zurückgreift. Die aktuelle Standard-Engine-Version ist 5.0.0 (Hinweis: Amazon DocumentDB 8.0 ist verfügbar, muss aber explizit angegeben werden). Wenn neue Hauptversionen der Engine veröffentlicht werden, `--engine-version` wird die Standard-Engine-Version für aktualisiert, sodass sie die letzte Hauptversion der Engine wiedergibt. Geben Sie daher für Produktions-Workloads und insbesondere für solche, die von Scripting, Automatisierung oder CloudFormation Vorlagen abhängig sind, explizit die gewünschte Hauptversion `--engine-version` an.

> **Note**
>
> Wenn ein `db-subnet-group-name` oder nicht angegeben `vpc-security-group-id` ist, verwendet Amazon DocumentDB die Standard-Subnetzgruppe und die Amazon VPC-Sicherheitsgruppe für die angegebene Region.

Für Linux, macOS oder Unix:

```
aws docdb create-db-cluster \
      --db-cluster-identifier sample-cluster \
      --engine docdb \
      --engine-version 5.0.0 \
      --deletion-protection \
      --master-username masteruser \
      --master-user-password password
```

Für Windows:

```
aws docdb create-db-cluster ^
      --db-cluster-identifier sample-cluster ^
      --engine docdb ^
      --engine-version 5.0.0 ^
      --deletion-protection ^
      --master-username masteruser ^
      --master-user-password password
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    "DBCluster": {
        "StorageEncrypted": false,
        "DBClusterMembers": [],
        "Engine": "docdb",
        "DeletionProtection" : "enabled",
        "ClusterCreateTime": "2018-11-26T17:15:19.885Z",
        "DBSubnetGroup": "default",
        "EngineVersion": "5.0.0",
        "MasterUsername": "masteruser",
        "BackupRetentionPeriod": 1,
        "DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster",
        "DBClusterIdentifier": "sample-cluster",
        "MultiAZ": false,
        "DBClusterParameterGroup": "default.docdb5.0",
        "PreferredBackupWindow": "09:12-09:42",
        "DbClusterResourceId": "cluster-KQSGI4MHU4NTDDRVNLNTU7XVAY",
        "PreferredMaintenanceWindow": "tue:04:17-tue:04:47",
        "Port": 27017,
        "Status": "creating",
```

```
        "ReaderEndpoint": "sample-cluster.cluster-ro-sfcrlcjcoroz.us-
east-1.docdb.amazonaws.com",
        "AssociatedRoles": [],
        "HostedZoneId": "ZNKXTT8WH85VW",
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-77186e0d",
                "Status": "active"
            }
        ],
        "AvailabilityZones": [
            "us-east-1a",
            "us-east-1c",
            "us-east-1e"
        ],
        "Endpoint": "sample-cluster.cluster-sfcrlcjcoroz.us-east-1.docdb.amazonaws.com"
    }
}
```

Die Erstellung des Clusters dauert mehrere Minuten. Sie können AWS-Managementkonsole oder verwenden, AWS CLI um den Status Ihres Clusters zu überwachen. Weitere Informationen finden Sie unter Überwachung des Status eines Amazon DocumentDB-Clusters.

> **Important**
>
> Wenn Sie den verwenden AWS CLI , um einen Amazon DocumentDB-Cluster zu erstellen, werden keine Instances erstellt. Daher müssen Sie explizit eine primäre Instance und alle benötigten Replikat-Instances anlegen. Sie können entweder die Konsole verwenden oder AWS CLI um die Instances zu erstellen. Weitere Informationen finden Sie unter Hinzufügen einer Amazon DocumentDB DocumentDB-Instance zu einem Cluster.

Weitere Informationen finden Sie unter CreateDBCluster in der Amazon DocumentDB-API-Referenz.

Beschreibung von Amazon DocumentDB-Clustern

Sie können entweder die Amazon DocumentDB Management Console oder die verwenden AWS CLI , um Details wie Verbindungsendpunkte, Sicherheitsgruppen, VPCs und Parametergruppen zu Ihren Amazon DocumentDB-Clustern anzuzeigen.

Weitere Informationen finden Sie hier:

- Überwachung des Status eines Amazon DocumentDB-Clusters
- Die Endpunkte eines Clusters finden

Using the AWS-Managementkonsole

Gehen Sie wie folgt vor, um die Details eines bestimmten Amazon DocumentDB-Clusters mithilfe der Konsole anzuzeigen.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Klicken Sie im Navigationsbereich auf Cluster.

   > **Tip**
   >
   > Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Wählen Sie in der Liste der Cluster den Namen des Clusters aus, dessen Details Sie sehen möchten. Die Informationen über den Cluster sind in die folgenden Gruppierungen unterteilt:

   - Zusammenfassung — Allgemeine Informationen zum Cluster, einschließlich der Engine-Version, des Cluster-Status, ausstehender Wartungsarbeiten und des Status der Parametergruppe.
   - Konnektivität und Sicherheit — Der Abschnitt Verbinden listet Verbindungsendpunkte auf, über die mit der Mongo-Shell oder mit einer Anwendung eine Verbindung zu diesem Cluster hergestellt werden kann. Im Abschnitt Sicherheitsgruppen werden die Sicherheitsgruppen aufgeführt, die diesem Cluster zugeordnet sind, sowie ihre VPC-ID und Beschreibungen.
   - Konfiguration — Im Abschnitt Cluster-Details sind Details zum Cluster aufgeführt, einschließlich des Amazon-Ressourcennamens (ARN), des Endpunkts und der Parametergruppe des Clusters. Außerdem werden die Sicherungsinformationen, Wartungsdetails sowie Sicherheits- und Netzwerkeinstellungen des Clusters aufgelistet. Im Abschnitt Cluster-Instances werden die Instances aufgeführt, die zu diesem Cluster gehören, mit dem Status der einzelnen Instance-Rollen und Cluster-Parametergruppen.

- Überwachung — Die Amazon CloudWatch Logs-Metriken für diesen Cluster. Weitere Informationen finden Sie unter Überwachung von Amazon DocumentDB mit CloudWatch.
- Ereignisse und Tags — Im Abschnitt Aktuelle Ereignisse werden die letzten Ereignisse für diesen Cluster aufgeführt. Amazon DocumentDB zeichnet Ereignisse auf, die sich auf Ihre Cluster, Instances, Snapshots, Sicherheitsgruppen und Cluster-Parametergruppen beziehen. Zu diesen Informationen gehören Datum, Uhrzeit und Nachricht, die jedem Ereignis zugeordnet sind. Der Abschnitt Tags listet die an diesen Cluster angehängten Tags auf.

Using the AWS CLI

Um die Details Ihrer Amazon DocumentDB-Cluster mithilfe von anzuzeigen AWS CLI, verwenden Sie den `describe-db-clusters` Befehl, wie in den folgenden Beispielen gezeigt. Weitere Informationen finden Sie unter DescribeDBClusters in der Amazon DocumentDB Resource Management API-Referenz.

> **Note**
>
> Für bestimmte Verwaltungsfunktionen wie das Cluster- und Instance-Lebenszyklusmanagement nutzt Amazon DocumentDB Betriebstechnologie, die mit Amazon RDS gemeinsam genutzt wird. Der `filterName=engine,Values=docdb` Filterparameter gibt nur Amazon DocumentDB-Cluster zurück.

Example

Beispiel 1: Alle Amazon DocumentDB-Cluster auflisten

Der folgende AWS CLI Code listet die Details für alle Amazon DocumentDB-Cluster in einer Region auf.

```
aws docdb describe-db-clusters --filter Name=engine,Values=docdb
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
    "DBClusters": [
        {
```

```
            "AvailabilityZones": [
                "us-east-1c",
                "us-east-1b",
                "us-east-1a"
            ],
            "BackupRetentionPeriod": 1,
            "DBClusterIdentifier": "sample-cluster-1",
            "DBClusterParameterGroup": "sample-parameter-group",
            "DBSubnetGroup": "default",
            "Status": "available",
            ...
        },
        {
            "AvailabilityZones": [
                "us-east-1c",
                "us-east-1b",
                "us-east-1a"
            ],
            "BackupRetentionPeriod": 1,
            "DBClusterIdentifier": "sample-cluster-2",
            "DBClusterParameterGroup": "sample-parameter-group",
            "DBSubnetGroup": "default",
            "Status": "available",
            ...
        },
        {
            "AvailabilityZones": [
                "us-east-1c",
                "us-east-1b",
                "us-east-1a"
            ],
            "BackupRetentionPeriod": 1,
            "DBClusterIdentifier": "sample-cluster-3",
            "DBClusterParameterGroup": "sample-parameter-group",
            "DBSubnetGroup": "default",
            "Status": "available",
            ...
        }
    ]
}
```

Example

Beispiel 2: Listet alle Details für einen bestimmten Amazon DocumentDB-Cluster auf

Der folgende AWS CLI Code listet die Details für den Cluster `sample-cluster` auf.

Für Linux, macOS oder Unix:

```
aws docdb describe-db-clusters \
   --filter Name=engine,Values=docdb \
   --db-cluster-identifier sample-cluster
```

Für Windows:

```
aws docdb describe-db-clusters ^
   --filter Name=engine,Values=docdb ^
   --db-cluster-identifier sample-cluster
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
    "DBClusters": [
        {
            "AllocatedStorage": 1,
            "AvailabilityZones": [
                "us-east-1c",
                "us-east-1a",
                "us-east-1d"
            ],
            "BackupRetentionPeriod": 2,
            "DBClusterIdentifier": "sample-cluster",
            "DBClusterParameterGroup": "sample-parameter-group",
            "DBSubnetGroup": "default",
            "Status": "available",
            "EarliestRestorableTime": "2023-11-07T22:34:08.148000+00:00",
            "Endpoint": "sample-cluster.node.us-east-1.amazon.com",
            "ReaderEndpoint": "sample-cluster.node.us-east-1.amazon.com",
            "MultiAZ": false,
            "Engine": "docdb",
            "EngineVersion": "5.0.0",
            "LatestRestorableTime": "2023-11-10T07:21:16.772000+00:00",
            "Port": 27017,
            "MasterUsername": "chimeraAdmin",
            "PreferredBackupWindow": "22:22-22:52",
            "PreferredMaintenanceWindow": "sun:03:01-sun:03:31",
            "ReadReplicaIdentifiers": [],
            "DBClusterMembers": [
```

```
                {
                    "DBInstanceIdentifier": "sample-instance-1",
                    "IsClusterWriter": true,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "PromotionTier": 1
                },
                {
                    "DBInstanceIdentifier": "sample-instance-2",
                    "IsClusterWriter": true,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "PromotionTier": 1
                },

            ],
            "VpcSecurityGroups": [
                {
                    "VpcSecurityGroupId": "sg-9084c2ec",
                    "Status": "active"
                }
            ],
            "HostedZoneId": "Z06853723JYKYBXTJ49RB",
            "StorageEncrypted": false,
            "DbClusterResourceId": "cluster-T4LGLANHVAPGQYYULWUDKLVQL4",
            "DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:sample-
cluster",
            "AssociatedRoles": [],
            "IAMDatabaseAuthenticationEnabled": false,
            "ClusterCreateTime": "2023-11-06T18:05:41.568000+00:00",
            "EngineMode": "provisioned",
            "DeletionProtection": false,
            "HttpEndpointEnabled": false,
            "CopyTagsToSnapshot": false,
            "CrossAccountClone": false,
            "DomainMemberships": [],
            "TagList": [],
            "StorageType": "iopt1",
            "AutoMinorVersionUpgrade": false,
            "NetworkType": "IPV4",
            "IOOptimizedNextAllowedModificationTime":
 "2023-12-07T18:05:41.580000+00:00"
        }
    ]
}
```

Example

Beispiel 3: Spezifische Details für einen Amazon DocumentDB-Cluster auflisten

Um eine Teilmenge der Cluster-Details mithilfe von aufzulisten AWS CLI, fügen Sie eine hinzu, die `--query` angibt, welche Cluster-Mitglieder der `describe-db-clusters` Vorgang auflisten soll. Der `--db-cluster-identifier`-Parameter ist der Bezeichner für den jeweiligen Cluster, dessen Details Sie anzeigen möchten. Weitere Informationen zu Abfragen finden Sie im Benutzerhandbuch unter So filtern Sie die Ausgabe mit der `--query` AWS Command Line Interface Option.

Das folgende Beispiel listet die Instances in einem Amazon DocumentDB-Cluster auf.

Für Linux, macOS oder Unix:

```
aws docdb describe-db-clusters \
    --filter Name=engine,Values=docdb \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterMembers]'
```

Für Windows:

```
aws docdb describe-db-clusters ^
    --filter Name=engine,Values=docdb ^
    --db-cluster-identifier sample-cluster ^
    --query 'DBClusters[*].[DBClusterMembers]'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
[
    [
        [
            {
                "DBInstanceIdentifier": "sample-instance-1",
                "IsClusterWriter": true,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            },
            {
                "DBInstanceIdentifier": "sample-instance-2",
                "IsClusterWriter": false,
```

```
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            }
        ]
    ]
]
```

Ändern eines Amazon DocumentDB-Clusters

Nur Cluster, die sich im Status available befinden, können geändert werden. Ein angehaltener Cluster kann nicht geändert werden. Wenn der Cluster angehalten ist, starten Sie zuerst den Cluster, warten Sie, bis der Cluster verfügbar wird, und nehmen Sie dann die gewünschten Änderungen vor. Weitere Informationen finden Sie unter Stoppen und Starten eines Amazon DocumentDB-Clusters.

Using the AWS-Managementkonsole

Gehen Sie wie folgt vor, um einen bestimmten Amazon DocumentDB-Cluster mithilfe der Konsole zu ändern.

So ändern Sie einen Amazon-DocumentDB-Cluster

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Klicken Sie im Navigationsbereich auf Cluster.

   > **Tip**
   >
   > Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Geben Sie den Cluster an, den Sie ändern möchten, indem Sie die Schaltfläche links neben dem Namen des Clusters auswählen.
4. Wählen Sie Actions (Aktionen) und dann Modify (Ändern) aus.
5. Nehmen Sie im Bereich Modify Cluster: <cluster-name> (Cluster ändern: <cluster-name>) die gewünschten Änderungen vor. Sie können in den folgenden Bereichen Änderungen durchführen:

- Cluster-Spezifikationen — Der Name, die Sicherheitsgruppen und die Verwaltung der Anmeldeinformationen des Clusters.
- Cluster-Speicherkonfiguration — Der Datenspeichermodus des Clusters. Wählen Sie zwischen Standard und I/O-Optimized Konfiguration.
- Cluster-Optionen — Der Port und die Parametergruppe des Clusters.
- Backup — Der Aufbewahrungszeitraum für Backups und das Backup-Fenster des Clusters.
- Protokollexporte — Aktiviert oder deaktiviert den Export von Audit- oder Profiler-Protokollen.
- Wartung — Legen Sie das Wartungsfenster des Clusters fest.
- Löschschutz — Aktiviert oder deaktiviert den Löschschutz auf dem Cluster. Der Löschschutz ist standardmäßig aktiviert.

6. Wenn Sie fertig sind, wählen Sie Continue (Weiter) aus, um eine Zusammenfassung Ihrer Änderungen anzuzeigen.
7. Wenn Sie mit den Änderungen zufrieden sind, können Sie Modify cluster (Cluster ändern) auswählen, um den Cluster zu ändern. Alternativ können Sie Back (Zurück) oder Cancel (Abbrechen) wählen, um die Änderungen zu bearbeiten oder abzubrechen.

Es dauert einige Minuten, bis Ihre Änderungen übernommen werden. Sie können den Cluster nur verwenden, wenn sein Status available ist. Sie können den Status des Clusters über die Konsole oder AWS CLIüberwachen. Weitere Informationen finden Sie unter Überwachung des Status eines Amazon DocumentDB-Clusters.

Using the AWS CLI

Verwenden Sie die `modify-db-cluster`-Operation, um den angegebenen Cluster mit der AWS CLI zu ändern. Weitere Informationen finden Sie unter ModifyDBCluster in der Amazon DocumentDB-API-Referenz.

Parameters

- **`--db-cluster-identifier`**—Erforderlich. Die ID des Amazon DocumentDB-Clusters, den Sie ändern werden.
- **`--backup-retention-period`**— Fakultativ. Die Anzahl von Tagen, über die automatische Backups aufbewahrt werden. Gültige Werte sind 1—35.

- **`--storage-type`**— Fakultativ. Die Speicherkonfiguration des Clusters. Gültige Werte sind `standard` (Standard) oder `iopt1` (I/O-optimized).
- **`--db-cluster-parameter-group-name`**— Fakultativ. Der Name der Cluster-Parametergruppe, die für den Cluster verwendet werden soll.
- **`--manage-master-user-password`**— Fakultativ. Amazon DocumentDB generiert das Masterbenutzerkennwort und verwaltet es während seines gesamten Lebenszyklus in Secrets Manager.
- **`--rotate-master-user-password`**— Fakultativ. Secrets Manager generiert eine neue geheime Version für das vorhandene Geheimnis. Die neue Version des Secrets enthält das neue primäre Benutzerkennwort. Amazon DocumentDB ändert das primäre Benutzerkennwort für den Cluster so, dass es dem Passwort für die neue geheime Version entspricht.

  Sie müssen die **`--apply-immediately`** Option angeben, wenn Sie das primäre Passwort ändern.
- **`--master-user-password`**— Fakultativ. Das neue Passwort für den primären Datenbankbenutzer.

  Passwortbeschränkungen:

  - Die Länge beträgt [8—100] druckbare ASCII-Zeichen.
  - Es können alle druckbaren ASCII-Zeichen mit Ausnahme der folgenden verwendet werden:
    - **`/`** (Schrägstrich)
    - **`"`** (doppeltes Anführungszeichen)
    - **`@`** ('At'-Symbol)
- **`--new-db-cluster-identifier`**— Fakultativ. Die neue Cluster-ID für den Cluster beim Umbenennen eines Clusters. Dieser Wert wird als Zeichenfolge in Kleinbuchstaben gespeichert.

  Benennungseinschränkungen:

  - Die Länge besteht aus [1—63] Buchstaben, Zahlen oder Bindestrichen.
  - Muss mit einem Buchstaben beginnen.
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.
  - Muss für alle Cluster in Amazon RDS, Amazon Neptune und Amazon DocumentDB pro Region eindeutig sein. AWS-Konto

- **`--preferred-backup-window`**— Fakultativ. Der tägliche Zeitbereich, in dem automatisierte Backups erstellt werden, in Universal Coordinated Time (UTC).
  - Format: `hh24:mm-hh24:mm`
- **`--preferred-maintenance-window`**— Fakultativ. Der wöchentliche Zeitbereich, in dem Systemwartungen stattfinden können, in UTC.
  - Format: `ddd:hh24:mm-ddd:hh24:mm`
  - Gültige Tage: `Sun`, `Mon`, `Tue`, `Wed`, `Thu`, `Fri` und `Sat`.
- **`--deletion-protection`**oder **`--no-deletion-protection`** —Optional. Gibt an, ob der Löschschutz für diesen Cluster aktiviert werden soll. Der Löschschutz verhindert, dass ein Cluster versehentlich gelöscht wird, solange der Cluster-Löschschutz nicht deaktiviert wurde. Weitere Informationen finden Sie unter Löschen eines Amazon DocumentDB-Clusters.
- **`--apply-immediately`**oder **`--no-apply-immediately`** — Verwenden Sie`--apply-immediately`, um die Änderung sofort vorzunehmen. Verwenden Sie `--no-apply-immediately`, um die Änderung während des nächsten Wartungsfensters Ihres Clusters vorzunehmen.

Example

Der folgende Code ändert die Aufbewahrungsfrist für das Backup für den Cluster `sample-cluster`.

Für Linux, macOS oder Unix:

```
aws docdb modify-db-cluster \
       --db-cluster-identifier sample-cluster \
       --apply-immediately \
       --backup-retention-period 7
```

Für Windows:

```
aws docdb modify-db-cluster ^
       --db-cluster-identifier sample-cluster ^
       --apply-immediately ^
       --backup-retention-period 7
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
```

```
    "DBCluster": {
        "BackupRetentionPeriod": 7,
        "DbClusterResourceId": "cluster-VDP53QEWST7YHM36TTXOPJT5YE",
        "Status": "available",
        "DBClusterMembers": [
            {
                "PromotionTier": 1,
                "DBClusterParameterGroupStatus": "in-sync",
                "DBInstanceIdentifier": "sample-cluster-instance",
                "IsClusterWriter": true
            }
        ],
        "ReadReplicaIdentifiers": [],
        "AvailabilityZones": [
            "us-east-1b",
            "us-east-1c",
            "us-east-1a"
        ],
        "ReaderEndpoint": "sample-cluster.cluster-ro-ctevjxdlur57.us-
east-1.rds.amazonaws.com",
        "DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster",
        "PreferredMaintenanceWindow": "sat:09:51-sat:10:21",
        "EarliestRestorableTime": "2018-06-17T00:06:19.374Z",
        "StorageEncrypted": false,
        "MultiAZ": false,
        "AssociatedRoles": [],
        "MasterUsername": "<your-master-user-name>",
        "DBClusterIdentifier": "sample-cluster",
        "VpcSecurityGroups": [
            {
                "Status": "active",
                "VpcSecurityGroupId": "sg-77186e0d"
            }
        ],
        "HostedZoneId": "Z2SUY0A1719RZT",
        "LatestRestorableTime": "2018-06-18T21:17:05.737Z",
        "AllocatedStorage": 1,
        "Port": 27017,
        "Engine": "docdb",
        "DBClusterParameterGroup": "default.docdb3.4",
        "Endpoint": "sample-cluster.cluster-ctevjxdlur57.us-
east-1.rds.amazonaws.com",
        "DBSubnetGroup": "default",
        "PreferredBackupWindow": "00:00-00:30",
```

```
        "EngineVersion": "3.4",
        "ClusterCreateTime": "2018-06-06T19:25:47.991Z",
        "IAMDatabaseAuthenticationEnabled": false
    }
}
```

Es dauert einige Minuten, bis Ihre Änderungen übernommen werden. Sie können den Cluster nur verwenden, wenn sein Status available ist. Sie können den Status des Clusters über die Konsole oder AWS CLIüberwachen. Weitere Informationen finden Sie unter Überwachung des Status eines Amazon DocumentDB-Clusters.

Feststellung ausstehender Wartungsarbeiten

Die Wartung umfasst in den meisten Fällen Aktualisierungen der folgenden Ressourcen in Ihrem Cluster:

- Betriebssystem (OS) der Instanzen im Cluster
- Engine-Version des Clusters

Weitere Informationen zur Wartung von Amazon DocumentDB finden Sie unterWartung von Amazon DocumentDB.

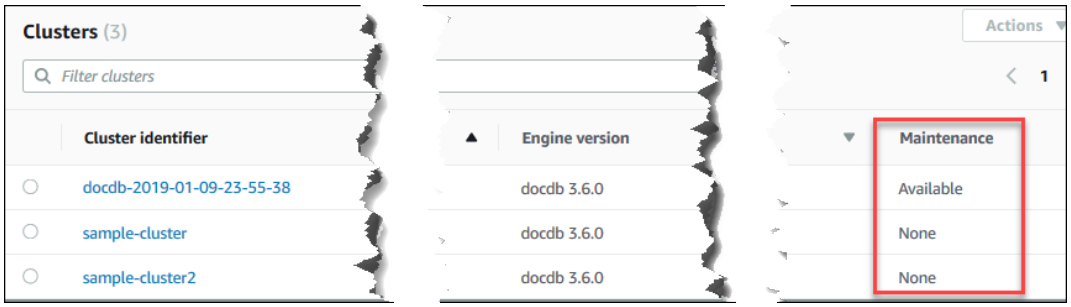
Using the AWS-Managementkonsole

Sie können das verwenden AWS-Managementkonsole , um festzustellen, ob für einen Cluster Wartungsarbeiten anstehen.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Klicken Sie im Navigationsbereich auf Cluster.

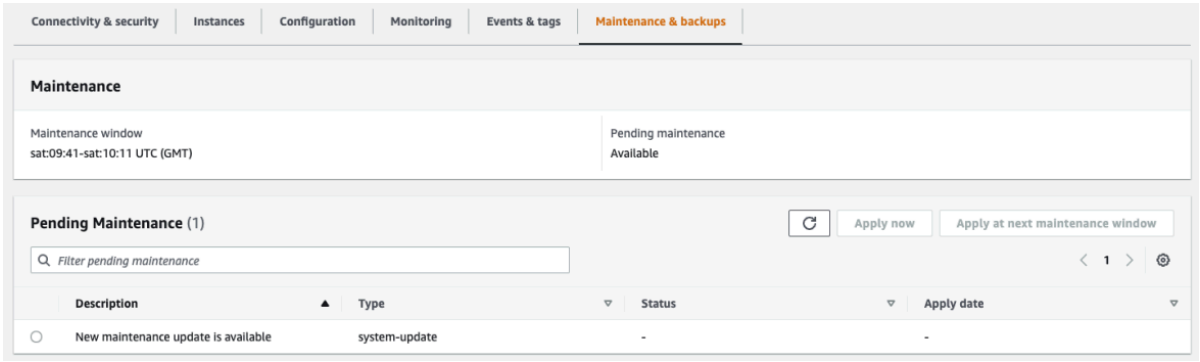
> **Tip**
>
> Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Überprüfen Sie die Spalte Maintenance (Wartung), um festzustellen, ob ein Cluster eine ausstehende Wartung hat.



Keine gibt an, dass auf dem Cluster die neueste Engine-Patch-Version ausgeführt wird und seine Instances über die neuesten Betriebssystemupdates verfügen. Verfügbar bedeutet, dass für den Cluster Wartungsarbeiten anstehen, was bedeuten kann, dass ein Engine-Upgrade oder ein Betriebssystem-Update erforderlich ist.

4. Wenn für Ihren Cluster Wartungsarbeiten ausstehen, können Sie die ausstehenden Wartungsmaßnahmen anzeigen, indem Sie den Clusternamen und dann die Registerkarte Wartung und Backups auswählen. Im Abschnitt „Ausstehende Wartung“ sind die ausstehenden Wartungsmaßnahmen aufgeführt.



Using the AWS CLI

Mithilfe des Vorgangs AWS CLI mit den folgenden Parametern können Sie ermitteln, ob ein Cluster über die neueste Engine-Version und die neuesten `describe-pending-maintenance-actions` Betriebssystemupdates verfügt.

Parameters

- **`--resource-identifier`**— Fakultativ. Der ARN der Ressource (Cluster). Wenn dieser Parameter weggelassen wird, werden ausstehende Wartungsaktionen für alle Cluster aufgelistet.

- **`--region`**— Fakultativ. Die AWS-Region, in der Sie diese Operation durchführen möchten (z. B. `us-east-1`).

Example

Für Linux, macOS oder Unix:

```
aws docdb describe-pending-maintenance-actions \
   --resource-identifier arn:aws:rds:aa-example-1:111122223333:cluster:sample-
cluster \
   --region aa-example-1
```

Für Windows:

```
aws docdb describe-pending-maintenance-actions ^
   --resource-identifier arn:aws:rds:aa-example-1:111122223333:cluster:sample-
cluster ^
   --region aa-example-1
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
    "PendingMaintenanceActions": [
        {
            "ResourceIdentifier": "arn:aws:rds:aa-
example-1:111122223333:cluster:sample-cluster",
            "PendingMaintenanceActionDetails": [
                {
                    "Description": "New feature",
                    "Action": "system-update",
                    "ForcedApplyDate": "2019-02-25T21:46:00Z",
                    "AutoAppliedAfterDate": "2019-02-25T07:41:00Z",
                    "CurrentApplyDate": "2019-02-25T07:41:00Z"
                }
            ]
        }
    ]
}
```

Die folgenden Wartungsaktionstypen gelten für Amazon DocumentDB-Cluster:

- `system-update`— Aktualisieren Sie den Engine-Patch für den Amazon DocumentDB-Cluster.

  Wenn für Ihren Cluster ein ausstehendes Engine-Patch-Upgrade verfügbar ist, fahren Sie mit den Schritten unter Durchführung eines Patch-Updates für die Engine-Version eines Clusters fort.
- `os-upgrade`— Aktualisieren Sie die Betriebssysteme aller DB-Instances im Amazon DocumentDB-Cluster mithilfe von fortlaufenden Upgrades.

  Weitere Informationen finden Sie unter Aktualisierungen des Amazon DocumentDB-Betriebssystems.

Durchführung eines Patch-Updates für die Engine-Version eines Clusters

In diesem Abschnitt erklären wir, wie Sie ein Patch-Update mithilfe von AWS-Managementkonsole oder bereitstellen AWS CLI. Ein Patch-Update ist ein Update innerhalb derselben Engine-Version (z. B. das Aktualisieren einer 3.6-Engine-Version auf eine neuere 3.6-Engine-Version). Sie können es sofort oder während des nächsten Wartungsfensters Ihres Clusters aktualisieren. Informationen dazu, ob Ihre Engine aktualisiert werden muss, finden Sie unterFeststellung ausstehender Wartungsarbeiten. Beachten Sie, dass Ihr Cluster einige Ausfallzeiten haben wird, wenn Sie das Update installieren.

> **Note**
>
> Wenn Sie versuchen, von einer Hauptversion der Engine auf eine andere zu aktualisieren, z. B. 3.6 auf 5.0 oder 5.0 auf 8.0, lesen Sie entweder Direktes Upgrade der Hauptversion von Amazon DocumentDB oderAktualisieren Ihres Amazon DocumentDB-Clusters mit AWS Database Migration Service.

Es gibt zwei Konfigurationsanforderungen, um die neuesten Patch-Updates für die Engine-Version eines Clusters zu erhalten:

- Der Cluster-Status muss available (verfügbar) lauten.
- Der Cluster muss eine ältere Engine-Version ausführen.

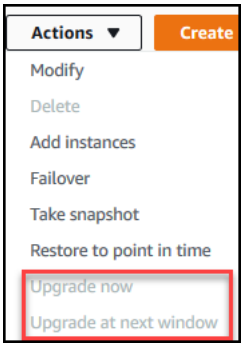
Using the AWS-Managementkonsole

Das folgende Verfahren wendet Patch-Updates mithilfe der Konsole auf die Engine-Version Ihres Clusters an. Sie haben die Möglichkeit, das Update sofort oder während des nächsten Wartungsfensters Ihres Clusters durchzuführen.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Klicken Sie im Navigationsbereich auf Cluster. Wählen Sie in der Liste der Cluster die Schaltfläche links neben dem Cluster aus, den Sie upgraden möchten. Der Cluster-Status muss available (verfügbar) lauten.

   > **Tip**
   >
   > Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Wählen Sie im Menü Actions (Aktionen) eine der folgenden Optionen aus. Diese Menüoptionen sind nur auswählbar, wenn der ausgewählte Cluster nicht die neueste Engine-Version ausführt.

   - Jetzt aktualisieren — Initiiert sofort den Upgrade-Prozess. Ihr Cluster ist eine zeitlang offline, während er auf die neueste Engine-Version aktualisiert wird.
   - Upgrade im nächsten Fenster — Initiiert den Upgrade-Vorgang im nächsten Wartungsfenster des Clusters. Ihr Cluster ist eine zeitlang offline, während er auf die neueste Engine-Version aktualisiert wird.
4. Wenn sich das Bestätigungsfenster öffnet, wählen Sie eine der folgenden Optionen aus:
   - Upgrade — Um Ihren Cluster gemäß dem im vorherigen Schritt ausgewählten Zeitplan auf die neueste Engine-Version zu aktualisieren.

- Abbrechen — Um das Engine-Upgrade des Clusters abzubrechen und mit der aktuellen Engine-Version des Clusters fortzufahren.

Using the AWS CLI

Sie können Patch-Updates auf Ihren Cluster anwenden, indem Sie den `apply-pending-maintenance-action` Vorgang AWS CLI und die folgenden Parameter verwenden.

Parameters

- **`--resource-identifier`**—Erforderlich. Der ARN des Amazon DocumentDB-Clusters, den Sie aktualisieren werden.
- **`--apply-action`**—Erforderlich. Wird verwendet`system-update`, um ein Engine-Patch-Update auf den Cluster anzuwenden.
- **`--opt-in-type`**—Erforderlich. Die folgenden Werte sind zulässig.
  - `immediate`— Wenden Sie die Wartungsaktion sofort an.
  - `next-maintenance`— Wenden Sie die Wartungsmaßnahme im nächsten Wartungsfenster an.
  - `undo-opt-in`— Stornieren Sie alle vorhandenen `next-maintenance` Opt-In-Anfragen.

Example

Im folgenden Beispiel wird das Engine-Patch-Update auf angewendet. `sample-cluster`

Für Linux, macOS oder Unix:

```
aws docdb apply-pending-maintenance-action \
   --resource-identifier arn:aws:rds:aa-example-1:111122223333:cluster:sample-
cluster \
   --apply-action system-update \
   --opt-in-type immediate
```

Für Windows:

```
aws docdb apply-pending-maintenance-action ^
   --resource-identifier arn:aws:rds:aa-example-1:111122223333:cluster:sample-
cluster ^
   --apply-action system-update ^
```

```
    --opt-in-type immediate
```

Die Ausgabe dieser Operation sieht folgendermaßen aus.

```
{
    "ResourcePendingMaintenanceActions": {
        "ResourceIdentifier": "arn:aws:rds:aa-example-1:111122223333:cluster:sample-
cluster",
        "PendingMaintenanceActionDetails": [
            {
                "CurrentApplyDate": "2019-02-20T20:57:06.904Z",
                "Description": "Bug fixes",
                "ForcedApplyDate": "2019-02-25T21:46:00Z",
                "OptInStatus": "immediate",
                "Action": "system-update",
                "AutoAppliedAfterDate": "2019-02-25T07:41:00Z"
            }
        ]
    }
}
```

Das Betriebssystem eines Clusters wird aktualisiert

Mithilfe der `os-upgrade` Wartungsaktion auf Clusterebene können Sie Betriebssystemupdates (OS) auf alle Instances in Ihrem Cluster anwenden. Amazon DocumentDB aktualisiert Instances fortlaufend, einige nacheinander, und aktualisiert die primäre Instance zuletzt, um Failovers zu minimieren. Das Update wird während des Cluster-Wartungsfensters ausgeführt. Nachdem eine Instanz ein Betriebssystem-Update erhalten hat, beginnt ihr Puffercache leer, und bei Abfragen auf dieser Instance kann es zu einer höheren Latenz kommen, bis das funktionierende Set wieder vom Speichervolume aufgefüllt wird.

Informationen dazu, ob für Ihren Cluster ein Betriebssystem-Update aussteht, finden Sie unterFeststellung ausstehender Wartungsarbeiten. Vollständige Informationen zu Betriebssystemaktualisierungen, einschließlich Konsolen- und CLI-Verfahren, finden Sie unterAktualisierungen des Amazon DocumentDB-Betriebssystems.

Stoppen und Starten eines Amazon DocumentDB-Clusters

Das Stoppen und Starten von Amazon DocumentDB-Clustern kann Ihnen helfen, die Kosten für Entwicklungs- und Testumgebungen zu verwalten. Anstatt jedes Mal, wenn Sie Amazon DocumentDB verwenden, Cluster und Instances zu erstellen und zu löschen, können Sie alle Instances in Ihrem Cluster vorübergehend anhalten, wenn sie nicht benötigt werden. Sie können sie neu starten, wenn Sie Ihre Tests fortsetzen.

Themen

- Übersicht über das Stoppen und Starten eines Clusters
- Operationen, die Sie auf einem gestoppten Cluster ausführen können

Übersicht über das Stoppen und Starten eines Clusters

In Zeiten, in denen Sie keinen Amazon DocumentDB-Cluster benötigen, können Sie alle Instances in diesem Cluster gleichzeitig beenden. Bei Bedarf können Sie den Cluster jederzeit erneut starten. Durch das Starten und Stoppen werden die Einrichtungs- und Entfernungsvorgänge für Cluster erleichtert, die in der Entwicklung, für Tests oder ähnliche Aktivitäten verwendet werden und keine kontinuierliche Verfügbarkeit erfordern. Sie können einen Cluster mithilfe von AWS-Managementkonsole oder AWS CLI mit einer einzigen Aktion beenden und starten, unabhängig davon, wie viele Instances sich im Cluster befinden.

Während Ihr Cluster gestoppt ist, bleibt des Cluster-Speichervolumen unverändert. Sie zahlen nur für Speicherung, manuelle Snapshots und automatischen Sicherungsspeicher innerhalb des angegebenen Aufbewahrungsfensters. Ihnen werden keine Instance-Stunden in Rechnung gestellt. Amazon DocumentDB startet Ihren Cluster automatisch nach sieben Tagen, sodass er nicht aufgrund der erforderlichen Wartungsupdates ausfällt. Wenn Ihr Cluster nach sieben Tagen wieder startet, werden Ihnen die Instances im Cluster wieder in Rechnung gestellt. Während Ihr Cluster gestoppt ist, können Sie Ihr Speichervolumen nicht abfragen, da die Abfrage erfordert, dass sich Instances im Status „available (verfügbar)“ befinden.

Wenn ein Amazon DocumentDB-Cluster gestoppt wird, können weder der Cluster noch seine Instances in irgendeiner Weise geändert werden. Dies umfasst das Hinzufügen oder Entfernen von Instances oder das Löschen des Clusters.

Using the AWS-Managementkonsole

Das folgende Verfahren zeigt, wie Sie einen Cluster mit einer oder mehreren Instances im Status „verfügbar“ stoppen oder einen gestoppten Cluster wieder starten.

Um einen Amazon DocumentDB-Cluster zu stoppen oder zu starten

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Klicken Sie im Navigationsbereich auf Cluster.

> **Tip**
>
> Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

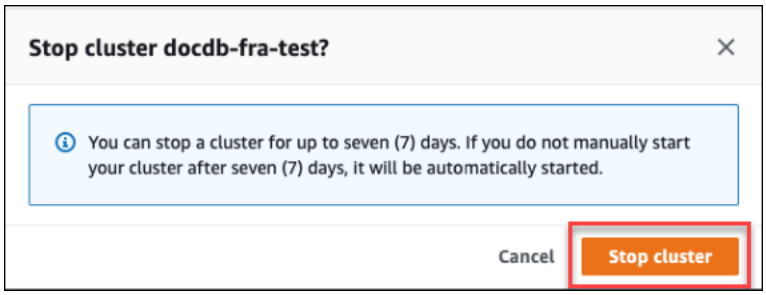
3. Wählen Sie in der Liste der Cluster die Schaltfläche links neben dem Namen des Clusters aus, den Sie stoppen oder starten möchten.
4. Wählen Sie Aktionen aus und dann die Aktion, die Sie auf dem Cluster ausführen möchten.
   - Wenn Sie den Cluster stoppen möchten und der Cluster verfügbar ist:

     a. Wählen Sie Beenden aus.

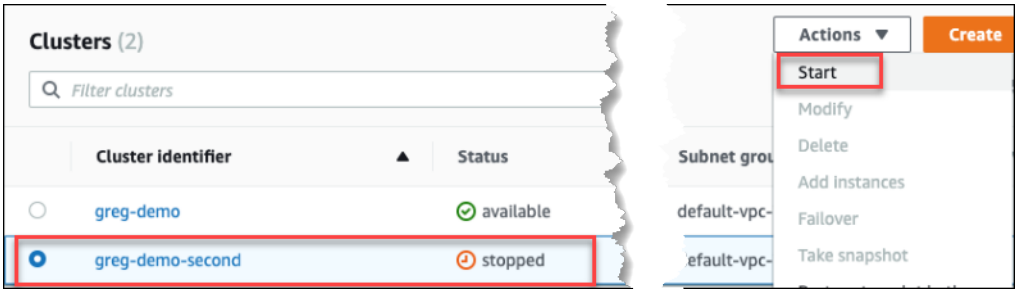


     Damit der Failover-Mechanismus nicht aktiviert wird, werden beim Stoppvorgang zuerst die Replikat-Instances und dann die primäre Instance gestoppt.

     b. Wenn Sie den Cluster stoppen möchten, wählen Sie im Bestätigungsdialogfeld Cluster stoppen aus und wenn der Cluster weiter ausgeführt werden soll, wählen Sie Abbrechen aus.



   - Wenn Sie einen gestoppten Cluster starten möchten, wählen Sie Starten aus.



5. Überwachen Sie den Status des Clusters und seiner Instances. Wenn Sie den Cluster gestartet haben, können Sie die Ausführung des Clusters fortsetzen, wenn der Cluster und dessen Instances verfügbar sind. Weitere Informationen finden Sie unter Den Status eines Clusters ermitteln.



Using the AWS CLI

Die folgenden Codebeispiele zeigen, wie Sie einen Cluster mit einer oder mehreren Instances im Status „verfügbar“ stoppen oder einen gestoppten Cluster starten.

Um einen Cluster mit einer oder mehreren verfügbaren Instances mithilfe von zu stoppen AWS CLI, verwenden Sie den `stop-db-cluster` Vorgang. Verwenden Sie die `start-db-cluster`-Operation, um einen gestoppten Cluster zu starten. Beide Operationen verwenden den `--db-cluster-identifier`-Parameter.

Parameter:

- **`--db-cluster-identifier`**—Erforderlich. Der Name des Clusters, der gestoppt oder gestartet werden soll.

Example— Um einen Cluster zu beenden, verwenden Sie den AWS CLI

Der folgende Code stoppt den Cluster `sample-cluster`. Der Cluster muss eine oder mehrere Instances im verfügbaren Zustand haben.

Für Linux, macOS oder Unix:

```
aws docdb stop-db-cluster \
   --db-cluster-identifier sample-cluster
```

Für Windows:

```
aws docdb stop-db-cluster ^
   --db-cluster-identifier sample-cluster
```

Example— Um einen Cluster mit dem zu starten AWS CLI

Der folgende Code startet den Cluster `sample-cluster`. Der Cluster muss jetzt gestoppt werden.

Für Linux, macOS oder Unix:

```
aws docdb start-db-cluster \
   --db-cluster-identifier sample-cluster
```

Für Windows:

```
aws docdb start-db-cluster ^
   --db-cluster-identifier sample-cluster
```

Operationen, die Sie auf einem gestoppten Cluster ausführen können

Während ein Amazon DocumentDB-Cluster gestoppt ist, können Sie eine Point-in-Time-Wiederherstellung zu einem beliebigen Zeitpunkt innerhalb des von Ihnen angegebenen Zeitfensters für die automatische Aufbewahrung von Backups durchführen. Weitere Informationen zur Durchführung einer zeitpunktbezogenen Wiederherstellung finden Sie unter Wiederherstellung auf einen bestimmten Zeitpunkt.

Sie können die Konfiguration eines Amazon DocumentDB-Clusters oder einer seiner Instances nicht ändern, solange der Cluster gestoppt ist. Es ist auch nicht möglich, Instances zum Cluster hinzuzufügen oder daraus zu entfernen oder den Cluster zu löschen, wenn ihm noch Instances zugeordnet sind. Vor solchen administrativen Aktionen müssen Sie den Cluster starten.

Amazon DocumentDB führt geplante Wartungsarbeiten auf Ihren gestoppten Cluster erst durch, nachdem er erneut gestartet wurde. Nach sieben Tagen startet Amazon DocumentDB automatisch

einen gestoppten Cluster, sodass er mit seinem Wartungsstatus nicht zu weit in Rückstand gerät. Sobald Ihr Cluster neu gestartet wird, werden Ihnen die Instances im Cluster wieder in Rechnung gestellt.

Während ein Cluster gestoppt wird, führt Amazon DocumentDB keine automatischen Backups durch und verlängert auch nicht die Aufbewahrungsfrist für Backups.

Löschen eines Amazon DocumentDB-Clusters

Sie können einen Amazon DocumentDB-Cluster mit dem AWS-Managementkonsole oder dem löschen. AWS CLI Um einen Cluster zu löschen, muss sich der Cluster im Status available (verfügbar) befinden und es dürfen keine Instances mit ihm verknüpft sein. Wenn der Cluster angehalten ist, starten Sie zuerst den Cluster, warten Sie, bis der Cluster verfügbar wird, und löschen Sie dann den Cluster. Weitere Informationen finden Sie unter Stoppen und Starten eines Amazon DocumentDB-Clusters.

Löschschutz

Zum Schutz Ihrer Cluster vor versehentlichem Löschen, können Sie den Löschschutz aktivieren. Wenn Sie mithilfe der Konsole einen Cluster erstellen, ist der Löschschutz standardmäßig aktiviert. Der Löschschutz ist jedoch standardmäßig deaktiviert, wenn Sie einen Cluster über die AWS CLI erstellen.

Amazon DocumentDB erzwingt den Löschschutz für einen Cluster, unabhängig davon, ob Sie den Löschvorgang über die Konsole oder die ausführen. AWS CLI Sie können einen Cluster nicht löschen, solange der Löschschutz aktiviert ist. Um einen Cluster mit aktiviertem Löschschutz zu löschen, müssen Sie zuerst den Cluster ändern und den Löschschutz deaktivieren.

Wenn Sie die Konsole mit aktiviertem Löschschutz für einen Cluster verwenden, können Sie die letzte Instance des Clusters nicht löschen, da dadurch auch der Cluster gelöscht werden würde. Sie können die letzte Instance eines mit Löschschutz geschützten Clusters mit der AWS CLI löschen. In diesen Fall bleibt der Cluster selbst bestehen und die Daten werden beibehalten. Sie können zum Zugriff auf die Daten neue Instances für den Cluster erstellen. Weitere Informationen zum Aktivieren und Deaktivieren des Löschschutzes siehe:

- Erstellen eines Amazon DocumentDB-Clusters
- Ändern eines Amazon DocumentDB-Clusters

Using the AWS-Managementkonsole

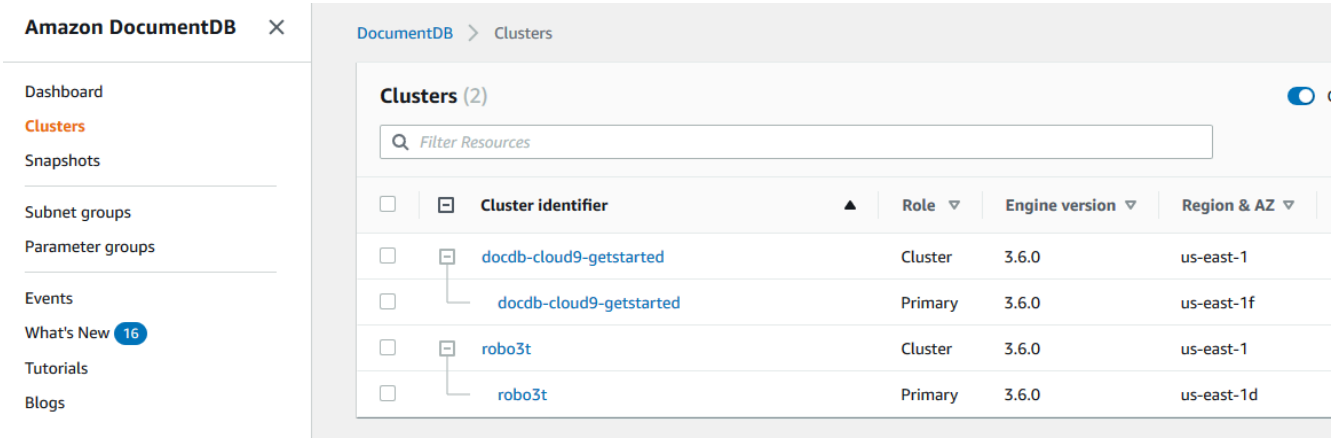
Um einen Cluster mithilfe von zu löschen AWS-Managementkonsole, muss der Löschschutz deaktiviert sein.

So finden Sie heraus, ob der Löschschutz für einen Cluster aktiviert ist:

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Klicken Sie im Navigationsbereich auf Cluster.

> **Tip**
>
> Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Beachten Sie, dass im Navigationsfeld Cluster in der Spalte Cluster-ID sowohl Cluster als auch Instances angezeigt werden. Instanzen werden unter Clustern aufgeführt, ähnlich wie im folgenden Screenshot.



4. Wählen Sie den Namen des Clusters und die Registerkarte Configuration (Konfiguration) aus. Suchen Sie im Abschnitt Cluster-Details die Option Deletion protection (Löschschutz). Wenn der Löschschutz aktiviert ist, bearbeiten Sie den Cluster, um den Löschschutz zu deaktivieren. Informationen über das Ändern eines Clusters finden Sie unter Ändern eines Amazon DocumentDB-Clusters.

   Nachdem Sie den Löschschutz deaktiviert haben, können Sie den Cluster löschen.

So löschen Sie einen Cluster:

1. Klicken Sie im Navigationsbereich auf Cluster.
2. Stellen Sie fest, ob der Cluster über Instanzen verfügt, indem Sie in der Spalte Cluster-ID nach den darunter aufgeführten Instances suchen. Bevor Sie einen Cluster löschen können, müssen Sie alle seine Instances löschen. Weitere Informationen finden Sie unter Löschen einer Amazon DocumentDB DocumentDB-Instance.
3. Abhängig davon, ob Ihr Cluster Instances hat oder nicht, führen Sie einen der folgenden Schritte aus:
   - Wenn der Cluster über keine Instances verfügt, wählen Sie die Schaltfläche links neben dem Cluster-Namen und dann Actions (Aktionen) aus. Wählen Sie im Dropdown-Menü Löschen aus. Füllen Sie das Dialogfeld <cluster-name> löschen aus und klicken Sie dann auf Löschen.
   - Wenn der Cluster eine oder mehrere Instances hat, gehen Sie wie folgt vor:
     a. Klicken Sie im Navigationsbereich auf Cluster.
     b. Löschen Sie jede Instanz des Clusters, indem Sie das Kontrollkästchen links neben dem Clusternamen auswählen. Wählen Sie Actions (Aktionen) und anschließend Delete (Löschen) aus. Füllen Sie das Dialogfeld <cluster-name> löschen aus und klicken Sie dann auf Löschen.

        Wenn Sie die letzte Instanz löschen, wird auch der Cluster gelöscht. Weitere Informationen zum Löschen von Instanzen finden Sie unterLöschen einer Amazon DocumentDB DocumentDB-Instance.

Es dauert einige Minuten, bis der Cluster gelöscht ist. Informationen zur Überwachung des Status des Clusters finden Sie unter Überwachung des Status eines Amazon DocumentDB-Clusters.

Using the AWS CLI

Sie können einen Cluster, dem Instances zugeordnet sind, nicht löschen. Um festzustellen, welche Instances mit dem Cluster verknüpft sind, führen Sie den Befehl `describe-db-clusters` aus und löschen Sie alle Instances des Clusters. Deaktivieren Sie dann bei Bedarf den Löschschutz auf Ihrem Cluster und löschen Sie schließlich den Cluster.

1. Löschen Sie zuerst alle Instances des Clusters.

Um festzustellen, welche Instances Sie löschen müssen, führen Sie den folgenden Befehl aus.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].
[DBClusterIdentifier,DBClusterMembers[*].DBInstanceIdentifier]'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
[
    [
        "sample-cluster",
        [
            "sample-instance-1",
            "sample-instance-2"
        ]
    ]
]
```

Wenn der Cluster, den Sie löschen möchten, Instances enthält, löschen Sie diese wie unten dargestellt.

```
aws docdb delete-db-instance \
    --db-instance-identifier sample-instance
```

2. Zweitens deaktivieren Sie den Löschschutz.

   Wenn Sie den verwenden AWS CLI , um alle Instanzen eines Clusters zu löschen, wird der Cluster nicht gelöscht. Sie müssen auch den Cluster löschen, aber dies ist nur möglich, wenn der Löschschutz deaktiviert ist.

   Um festzustellen, ob der Löschschutz für den Cluster aktiviert ist, führen Sie den folgenden Befehl aus.

> **Tip**
>
> Um den Löschschutzstatus all Ihrer Amazon DocumentDB-Cluster zu sehen, lassen Sie den `--db-cluster-identifier` Parameter weg.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterIdentifier,DeletionProtection]'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
[
    [
        "sample-cluster",
        "true"
    ]
]
```

Wenn für einen Cluster der Löschschutz aktiviert ist, bearbeiten Sie den Cluster, um den Löschschutz zu deaktivieren. Führen Sie den folgenden Befehl aus, um den Löschschutz des Clusters zu deaktivieren.

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --no-deletion-protection \
    --apply-immediately
```

3. Löschen Sie schließlich den Cluster.

   Nachdem Sie den Löschschutz deaktiviert haben, können Sie den Cluster löschen. Um einen Cluster zu löschen, verwenden Sie die `delete-db-cluster`-Operation mit den folgenden Parametern.

   - **`--db-cluster-identifier`**—Erforderlich. Die ID des Clusters, den Sie löschen möchten.

- **`--final-db-snapshot-identifier`**— Fakultativ. Wenn Sie einen endgültigen Snapshot wünschen, müssen Sie diesen Parameter mit einem Namen für den endgültigen Snapshot angeben. Sie müssen entweder `--final-db-snapshot-identifier` oder `--skip-final-snapshot` angeben.

  Benennungseinschränkungen:

  - Die Länge besteht aus [1—63] Buchstaben, Zahlen oder Bindestrichen.
  - Muss mit einem Buchstaben beginnen.
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.
  - Muss für alle Cluster in Amazon RDS, Amazon Neptune und Amazon DocumentDB pro Region eindeutig sein. AWS-Konto
- **`--skip-final-snapshot`**— Fakultativ. Verwenden Sie diesen Parameter nur, wenn Sie vor dem Löschen Ihres Clusters keinen letzten Snapshot machen möchten. Standardmäßig wird ein letzter Snapshot erstellt. Sie müssen entweder `--final-db-snapshot-identifier` oder `--skip-final-snapshot` angeben.

Der folgende AWS CLI Code löscht den Cluster `sample-cluster` mit einem endgültigen Snapshot. Die Operation schlägt fehl, wenn Instances vorhanden sind, die mit dem Cluster verknüpft sind, oder wenn der Löschschutz aktiviert ist.

Example

Für Linux, macOS oder Unix:

```
aws docdb delete-db-cluster \
    --db-cluster-identifier sample-cluster \
    --final-db-snapshot-identifier sample-cluster-final-snapshot
```

Für Windows:

```
aws docdb delete-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --final-db-snapshot-identifier sample-cluster-final-snapshot
```

Example

Der folgende AWS CLI Code löscht den Cluster, `sample-cluster` ohne einen endgültigen Snapshot zu erstellen.

Für Linux, macOS oder Unix:

```
aws docdb delete-db-cluster \
    --db-cluster-identifier sample-cluster \
    --skip-final-snapshot
```

Für Windows:

```
aws docdb delete-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --skip-final-snapshot
```

Die Ausgabe der `delete-db-cluster`-Operation ist der Cluster, den Sie löschen.

Es dauert einige Minuten, bis der Cluster gelöscht ist. Informationen zur Überwachung des Status des Clusters finden Sie unter Überwachung des Status eines Clusters.

Skalieren von Amazon DocumentDB-Clustern

Amazon DocumentDB ermöglicht es Ihnen, den Speicher und die Rechenleistung in Ihren Clustern nach Ihren Bedürfnissen zu skalieren. In diesem Abschnitt wird beschrieben, wie Sie Speicherskalierung, Instance-Skalierung und Leseskalierung verwenden können, um die Leistung und Skalierung Ihrer Amazon DocumentDB-Cluster und Instances zu verwalten.

Themen

- Speicherskalierung
- Skalierung von Instances
- Skalierung von Lesevorgängen
- Skalierung schreiben

Speicherskalierung

Der Amazon DocumentDB-Speicher skaliert automatisch mit den Daten in Ihrem Cluster-Volume. Wenn Ihre Daten wachsen, wächst Ihr Cluster-Volume-Speicher in Schritten von 10 GiB, auf bis zu 256 TiB für Engine-Version 8.0 und höher (128 TiB für frühere Engine-Versionen).

Skalierung von Instances

Sie können Ihren Amazon DocumentDB-Cluster nach Bedarf skalieren, indem Sie die Instance-Klasse für jede Instance im Cluster ändern. Amazon DocumentDB unterstützt mehrere Instance-Klassen, die für Amazon DocumentDB optimiert sind.

Weitere Informationen finden Sie unter Ändern einer Amazon DocumentDB DocumentDB-Instance.

Skalierung von Lesevorgängen

Sie können eine Leseskalierung für Ihren Amazon DocumentDB-Cluster erreichen, indem Sie bis zu 15 Amazon DocumentDB-Repliken im Cluster erstellen. Jedes Amazon DocumentDB-Replikat gibt dieselben Daten aus dem Cluster-Volume mit minimaler Replikatverzögerung zurück — in der Regel weniger als 100 Millisekunden, nachdem die primäre Instance ein Update geschrieben hat. Wenn Ihr Lesetraffic zunimmt, können Sie zusätzliche Amazon DocumentDB-Repliken erstellen und sich direkt mit ihnen verbinden, um die Leselast für Ihren Cluster zu verteilen. Amazon DocumentDB-Repliken müssen nicht derselben Instance-Klasse wie die primäre Instance angehören.

Weitere Informationen finden Sie unter Hinzufügen einer Amazon DocumentDB DocumentDB-Instance zu einem Cluster.

Um die Leseskala mit Amazon DocumentDB zu nutzen, stellen Sie als Replikatgruppe eine Verbindung zu Ihrem Cluster her und verteilen Sie Lesevorgänge mithilfe der integrierten Lesepräferenzfunktionen Ihres Treibers an Replica-Instances. Weitere Informationen finden Sie unter Verbindung zu Amazon DocumentDB als Replikatsatz herstellen.

Skalierung schreiben

Sie können die Schreibkapazität in Ihrem Amazon DocumentDB-Cluster skalieren, indem Sie die Größe der primären Instance Ihres Clusters erhöhen. In diesem Abschnitt werden zwei Methoden für die Skalierung der primären Instance Ihres Clusters abhängig von Ihren Anforderungen beschrieben. Die erste Option zielt darauf ab, die Auswirkungen auf die Anwendung zu minimieren, erfordert jedoch mehr Schritte, um abgeschlossen zu werden. Die zweite Option ermöglicht Ihnen die

Optimierung in Bezug auf die Einfachheit, da sie weniger Schritte erfordert. Die potenziellen Auswirkungen auf Anwendungen sind jedoch größer.

Abhängig von Ihrer Anwendung können Sie wählen, welcher Ansatz für Sie am besten geeignet ist. Weitere Informationen zu den verfügbaren Instance-Größen und Kosten finden Sie auf der Preisseite von Amazon DocumentDB.

1. Optimieren Sie für hohe Verfügbarkeit und Leistung — Wenn Sie im Replikatsatzmodus eine Verbindung zu Ihrem Cluster herstellen (empfohlen), können Sie das folgende Verfahren verwenden, um die Auswirkungen auf Ihre Anwendung bei der Skalierung Ihrer primären Instance zu minimieren. Diese Methode minimiert die Auswirkungen, da Ihr Cluster den von Ihnen festgelegten Wert für Hochverfügbarkeit weiter einhält oder überschreitet. Darüber hinaus werden Leseskalierungsziele dem Cluster als Instances hinzugefügt, anstatt an Ort und Stelle aktualisiert zu werden.

   a. Fügen Sie ein oder mehrere Replikate des größeren Instance-Typs zu Ihrem Cluster hinzu (siehe ???). Stellen Sie sicher, dass alle Replikate vom gleichen oder einem größeren Instance-Typ wie die primäre Instanz sind. Dies vermeidet eine unbeabsichtigte Reduzierung der Schreibleistung aufgrund eines Failovers zu einem kleineren Instance-Typ. In den meisten Fällen bedeutet dies, dass Sie die Anzahl der Instances in Ihrem Cluster vorübergehend verdoppeln und die kleineren Replikate dann nach Abschluss der Skalierung entfernen.

   b. Legen Sie die Failover-Stufe für alle neuen Replikate auf die Priorität Null fest. Dies stellt sicher, dass ein Replikat des kleineren Instance-Typs die höchste Failover-Priorität besitzt. Weitere Informationen finden Sie unter ???.

   c. Initiieren Sie einen manuellen Failover. Hierdurch wird eines der neuen Replikate zur primären Instance hochgestuft. Weitere Informationen finden Sie unter ???.

      > **Note**
      >
      > Dies führt zu einer Ausfallzeit von ca. 30 Sekunden für Ihren Cluster. Planen Sie diese Ausfallzeit ein.

   d. Entfernen Sie aus dem Cluster alle Replikate von Instance-Typen, die kleiner als der neue primäre Instance-Typ sind.

   e. Setzen Sie die Failover-Stufe aller Instances auf dieselbe Priorität zurück. (In der Regel bedeutet dies, dass sie auf 1 zurückgesetzt werden.)

Angenommen, Sie führen einen Cluster aus, der zurzeit drei `r5.large`-Instances enthält (eine primäre Instance und zwei Replikate). Sie möchten diese Instances auf den Instance-Typ `r5.xlarge` skalieren. Hierzu fügen Sie dem Cluster zunächst drei `r5.xlarge`-Replikat-Instances hinzu und legen anschließend die Failover-Stufe der neuen `r5.xlarge`-Replikate auf null fest. Als Nächstes leiten Sie einen manuellen Failover ein. (Ihnen ist bekannt, dass dies zu einer Ausfallzeit von ca. 30 Sekunden für Ihre Anwendung führt.) Nach Abschluss des Failovers entfernen Sie alle drei `r5.large`-Instances aus dem Cluster. Der Cluster ist nun auf `r5.xlarge`-Instances skaliert.

Um die Kosten zu optimieren, werden Amazon DocumentDB-Instances in Sekundenschritten in Rechnung gestellt, wobei nach einer fakturierbaren Statusänderung wie dem Erstellen, Ändern oder Löschen einer Instance eine Mindestgebühr von zehn Minuten anfällt. Weitere Informationen finden Sie unter Kostenoptimierung in der Dokumentation zu bewährten Methoden.

2. Aus Gründen der Einfachheit optimieren — Dieser Ansatz optimiert aus Gründen der Einfachheit. Dadurch wird der Cluster nicht erweitert und zusammengezogen, aber Ihre Lesekapazität kann dadurch vorübergehend reduziert werden.

   Es ist möglich, dass eine Änderung der Instance-Klasse eines Replikats dazu führt, dass diese Instance für einen kurzen Zeitraum, von einigen Sekunden bis weniger als 30 Sekunden, keine Anfragen bearbeitet. Wenn Sie im Replikatsatzmodus eine Verbindung zu Ihrem Cluster herstellen (empfohlen), würde dies Ihre Lesekapazität während des Skalierungsvorgangs um ein Replikat reduzieren (z. B. auf 66% Kapazität in einem Cluster mit 3 Knoten oder 75% Kapazität in einem Cluster mit 4 Knoten usw.).

   a. Skalieren Sie eine der Replikatinstanzen in Ihrem Cluster. Weitere Informationen finden Sie unter Instanzklassen verwalten.

   b. Warten Sie, bis die Instanz verfügbar ist (siehe Überwachung des Status einer Amazon DocumentDB-Instance).

      > **Note**
      >
      > Dies führt zu einer Ausfallzeit von ca. 30 Sekunden für Ihren Cluster. Planen Sie diese Ausfallzeit ein.

c. Führen Sie die Schritte 1 und 2 weiter aus, bis alle Replikatinstanzen nacheinander skaliert wurden.

d. Initiieren Sie ein manuelles Failover. Dadurch wird eines der Replikate zur primären Instanz hochgestuft. Weitere Informationen finden Sie unter Amazon DocumentDB-Failover.

> **Note**
>
> Dies führt zu einer Ausfallzeit von bis zu 30 Sekunden für Ihren Cluster, nimmt jedoch häufig weniger Zeit in Anspruch. Planen Sie diese Ausfallzeit ein.

e. Skalieren Sie die frühere primäre Instanz (jetzt eine Replikat-) Instance.

Klonen eines Volumes für einen Amazon DocumentDB-Cluster

Mithilfe des Amazon DocumentDB-Clonings können Sie einen neuen Cluster erstellen, der dasselbe Amazon DocumentDB-Cluster-Volume verwendet und über dieselben Daten wie das Original verfügt. Der Prozess ist so konzipiert, dass er schnell und kostengünstig ist. Der neue Cluster mit dem zugehörigen Datenvolume wird als clone (Klon) bezeichnet. Das Erstellen eines Klons ist schneller und platzsparender als das physische Kopieren der Daten mit anderen Techniken, wie z. B. das Wiederherstellen eines Snapshots.

Amazon DocumentDB unterstützt die Erstellung eines von Amazon DocumentDB bereitgestellten Klons aus einem bereitgestellten Amazon DocumentDB-Cluster. Wenn Sie einen Klon mit einer anderen Bereitstellungskonfiguration als die Quelle erstellen, wird der Klon mit der neuesten Version der Amazon DocumentDB-Engine der Quelle erstellt.

Wenn Sie Klone aus Ihren Amazon DocumentDB-Clustern erstellen, werden die Klone in Ihrem Konto erstellt AWS — dem gleichen Konto, dem der Amazon DocumentDB-Quellcluster gehört.

Themen

- Überblick über das Klonen von Amazon DocumentDB
- Nutzungshinweise für das Klonen von Amazon DocumentDB
- So funktioniert das Klonen von Amazon DocumentDB
- Einen Amazon DocumentDB-Klon erstellen

Überblick über das Klonen von Amazon DocumentDB

Amazon DocumentDB verwendet ein Copy-on-Write-Protokoll, um einen Klon zu erstellen. Dieser Mechanismus verwendet minimalen zusätzlichen Speicherplatz, um einen ersten Klon zu erstellen. Wenn der Clone zum ersten Mal erstellt wird, speichert Amazon DocumentDB eine einzige Kopie der Daten, die vom Quell-DB-Cluster und dem neuen (geklonten) Amazon DocumentDB-Cluster verwendet werden. Zusätzlicher Speicher wird nur zugewiesen, wenn Änderungen an Daten (auf dem Amazon DocumentDB-Speichervolume) durch den Amazon DocumentDB-Quell-Cluster oder den Amazon DocumentDB-Cluster-Klon vorgenommen werden. Weitere Informationen zum Copy-on-Write-Protokoll finden Sie unter So funktioniert das Klonen von Amazon DocumentDB.

Das Klonen von Amazon DocumentDB ist besonders nützlich, um schnell Testumgebungen mit Ihren Produktionsdaten einzurichten, ohne das Risiko einer Datenbeschädigung einzugehen. Sie können Klone für viele Arten von Anwendungen verwenden, z. B. für Folgende:

- Experimentieren Sie mit möglichen Änderungen (z. B. Schemaänderungen und Parametergruppenänderungen), um alle Auswirkungen zu bewerten.
- Führen Sie Workload-intensive Vorgänge aus, z. B. das Exportieren von Daten oder das Ausführen analytischer Abfragen auf dem Klon.
- Erstellen Sie eine Kopie Ihres Produktions-DB-Clusters zu Entwicklungs-, Test- oder anderen Zwecken.

Sie können mehr als einen Klon aus demselben Amazon DocumentDB-Cluster erstellen. Sie können auch mehrere Klone aus einem anderen Klon erstellen.

Nachdem Sie einen Amazon DocumentDB-Klon erstellt haben, können Sie die Amazon DocumentDB-Instances anders als den Amazon DocumentDB-Quell-Cluster konfigurieren. Beispielsweise benötigen Sie möglicherweise keinen Klon für Entwicklungszwecke, um dieselben Hochverfügbarkeitsanforderungen wie der Amazon DocumentDB-Quell-Produktionscluster zu erfüllen. In diesem Fall können Sie den Clone mit einer einzigen Amazon DocumentDB-Instance konfigurieren, anstatt mit mehreren DB-Instances, die vom Amazon DocumentDB-Cluster verwendet werden.

Wenn Sie den Klon für Test-, Entwicklungs- oder andere Zwecke nicht mehr verwenden, können Sie ihn löschen.

Nutzungshinweise für das Klonen von Amazon DocumentDB

Amazon DocumentDB; das Klonen hat derzeit die folgenden Einschränkungen:

- Sie können so viele Klone erstellen, wie Sie möchten, bis zur maximalen Anzahl von DB-Clustern, die in der AWS-Region zulässig sind. Nachdem Sie jedoch 15 Klone erstellt haben, ist der nächste Klon eine vollständige Kopie. Der Klonvorgang funktioniert wie eine zeitpunktbezogene Wiederherstellung.
- Sie können keinen Clone in einer anderen AWS Region als dem Amazon DocumentDB-Quell-Cluster erstellen.
- Sie können keinen Clone aus einem Amazon DocumentDB-Cluster erstellen, der keine DB-Instances hat. Sie können nur Amazon DocumentDB-Cluster klonen, die über mindestens eine DB-Instance verfügen.
- Sie können einen Klon in einer anderen Virtual Private Cloud (VPC) als der des Amazon DocumentDB-Clusters erstellen. In diesem Fall müssen die Subnetze der VPCs denselben Availability Zones zugeordnet sein.

So funktioniert das Klonen von Amazon DocumentDB

Das Klonen von Amazon DocumentDB funktioniert auf der Speicherebene eines Amazon DocumentDB-Clusters. Es verwendet ein Copy-on-Write-Protokoll, das sowohl schnell als auch platzsparend ist, was die zugrunde liegenden dauerhaften Medien angeht, die das Amazon DocumentDB-Speichervolume unterstützen. Weitere Informationen zu Amazon DocumentDB-Cluster-Volumes finden Sie unter. Verwaltung von Amazon DocumentDB-Clustern

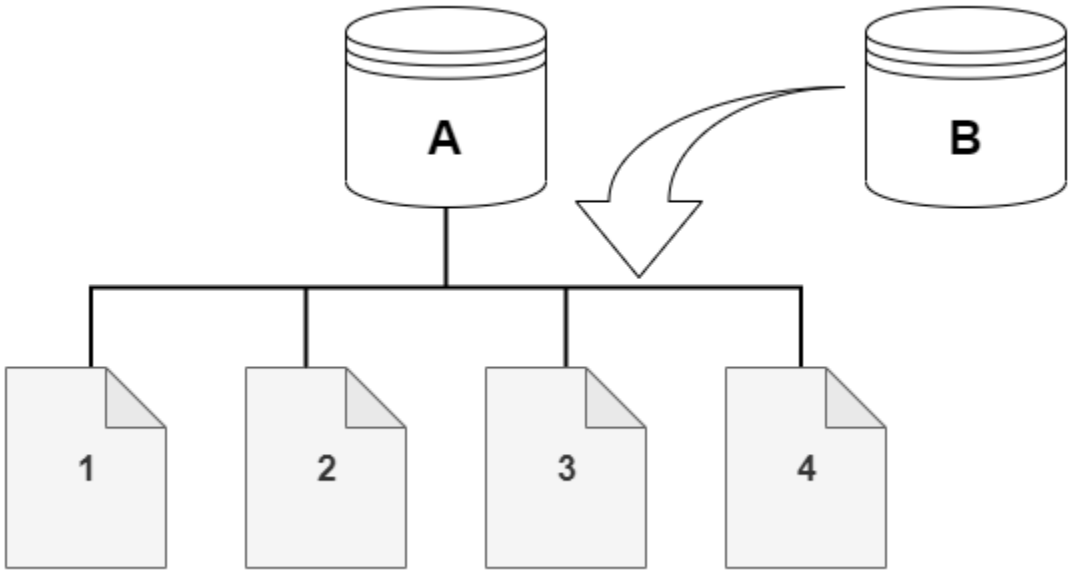
Themen

- Copy-On-Write-Protokoll verstehen
- Löschen eines Quell-Cluster-Volumes

Copy-On-Write-Protokoll verstehen

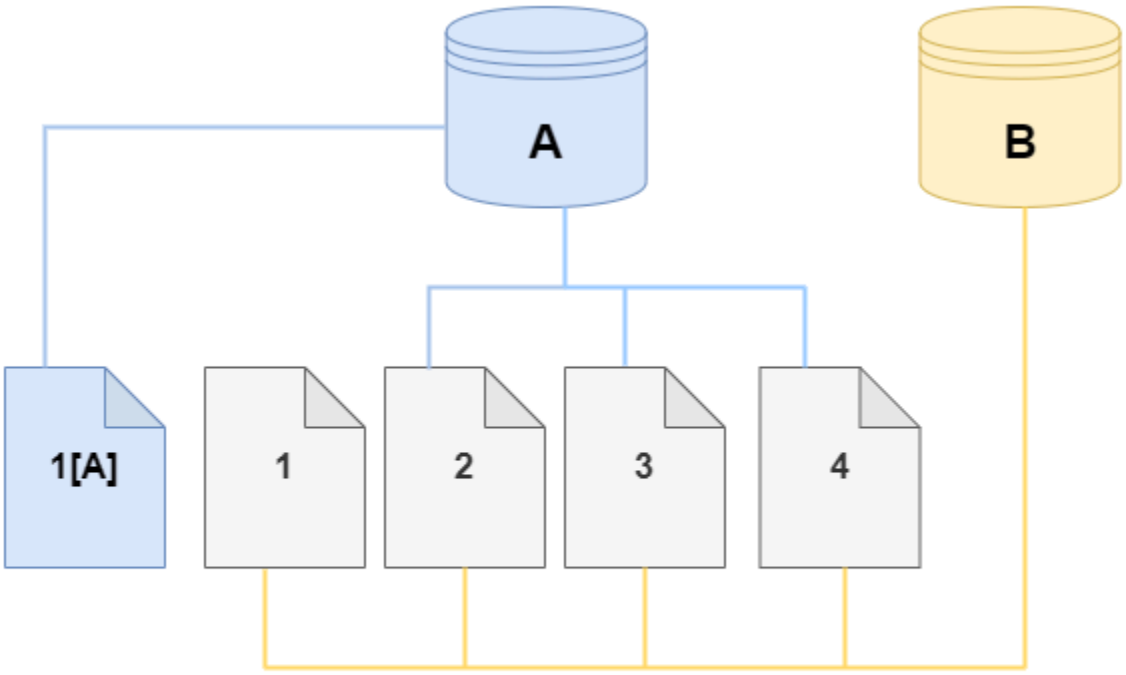
Ein Amazon DocumentDB-Cluster speichert Daten auf Seiten im zugrunde liegenden Amazon DocumentDB-Speichervolume.

Im folgenden Diagramm finden Sie beispielsweise einen Amazon DocumentDB-Cluster (A) mit vier Datenseiten (1, 2, 3 und 4). Stellen Sie sich vor, ein Klon, B, wird aus dem Amazon DocumentDB-Cluster erstellt. Wenn der Klon erstellt wird, werden keine Daten kopiert. Vielmehr verweist der Klon auf denselben Seitensatz wie der Amazon DocumentDB-Quellcluster.

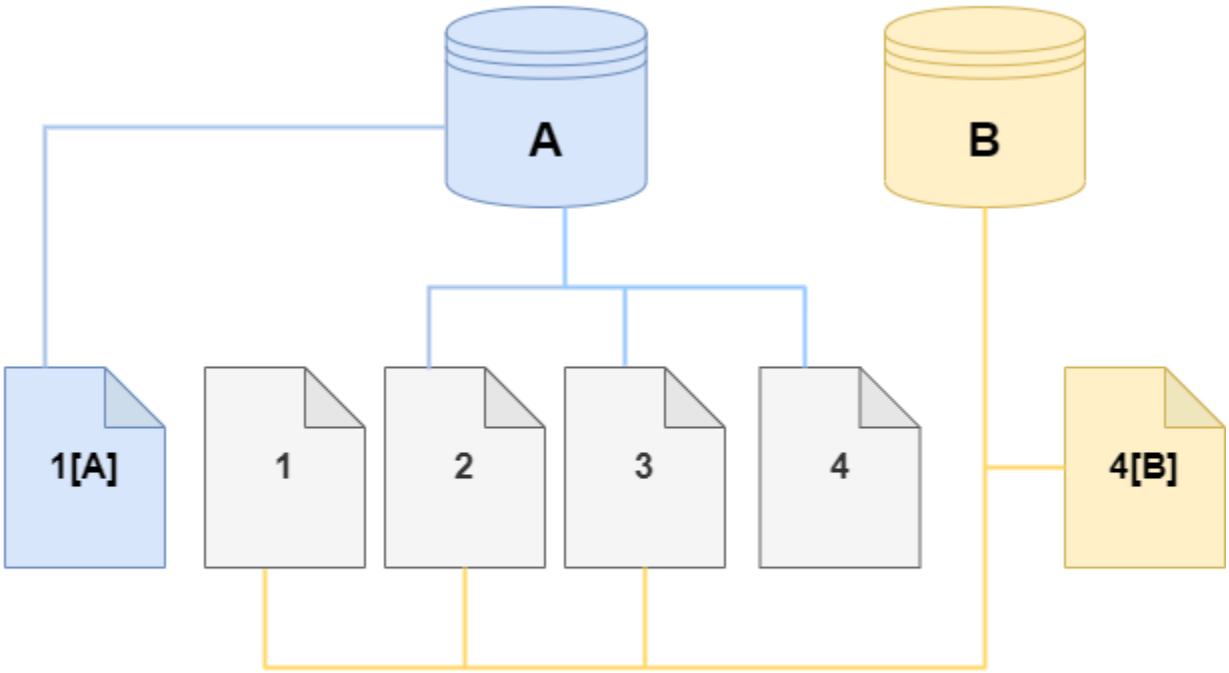


Wenn der Klon erstellt wird, ist normalerweise kein zusätzlicher Speicher erforderlich. Das Copy-on-Write-Protokoll verwendet dasselbe Segment auf dem physischen Speichermedium wie das Quellsegment. Zusätzlicher Speicher ist nur erforderlich, wenn die Kapazität des Quellsegments für das gesamte Klonsegment nicht ausreicht. Wenn dies der Fall ist, wird das Quellsegment auf ein anderes physisches Gerät kopiert.

In den folgenden Diagrammen finden Sie ein Beispiel für das Copy-on-Write-Protokoll in Aktion mit demselben Cluster A und dessen Klon, B, wie oben gezeigt. Nehmen wir an, Sie nehmen eine Änderung an Ihrem Amazon DocumentDB-Cluster (A) vor, die zu einer Änderung der Daten auf Seite 1 führt. Anstatt auf die ursprüngliche Seite 1 zu schreiben, erstellt Amazon DocumentDB eine neue Seite 1 [A]. Das Amazon DocumentDB-Cluster-Volume für Cluster (A) zeigt jetzt auf Seite 1 [A], 2, 3 und 4, während der Clone (B) immer noch auf die Originalseiten verweist.



Auf dem Klon wird eine Änderung an Seite 4 auf dem Speichervolume vorgenommen. Anstatt auf die ursprüngliche Seite 4 zu schreiben, erstellt Amazon DocumentDB eine neue Seite, 4 [B]. Der Klon verweist nun auf die Seiten 1, 2, 3 und auf Seite 4[B], während der Cluster (A) weiterhin auf 1[A], 2, 3 und 4 verweist.



Da im Laufe der Zeit sowohl im Amazon DocumentDB-Cluster-Quellvolume als auch im Clone immer mehr Änderungen vorgenommen werden, wird mehr Speicherplatz benötigt, um die Änderungen zu erfassen und zu speichern.

Löschen eines Quell-Cluster-Volumes

Wenn Sie ein Quell-Cluster-Volume löschen, dem ein oder mehrere Klone zugeordnet sind, sind die Klone nicht betroffen. Die Klone verweisen weiter auf die Seiten, die zuvor im Besitz des Quell-Cluster-Volumes waren.

Einen Amazon DocumentDB-Klon erstellen

Sie können einen Clone in demselben AWS Konto wie den Amazon DocumentDB-Quell-Cluster erstellen. Dazu können Sie das AWS-Managementkonsole oder das AWS CLI und die folgenden Verfahren verwenden.

Mithilfe des Amazon DocumentDB-Clonings können Sie einen bereitgestellten Amazon DocumentDB-Cluster-Klon aus einem bereitgestellten Amazon DocumentDB-Cluster erstellen.

Using the AWS-Managementkonsole

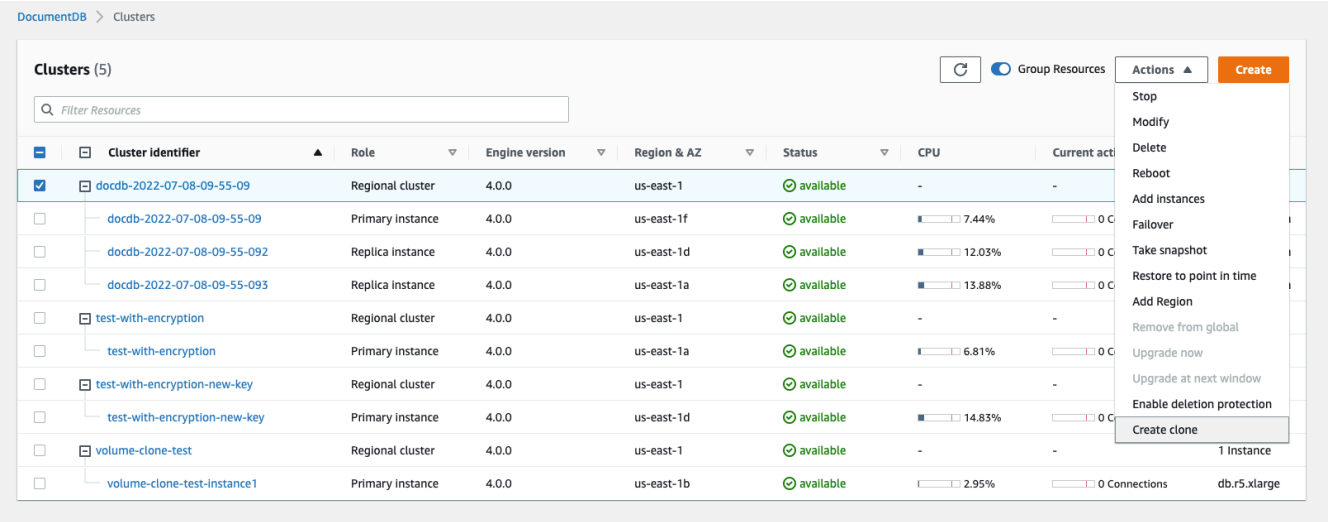
Das folgende Verfahren beschreibt, wie Sie einen Amazon DocumentDB-Cluster mithilfe von klonen. AWS-Managementkonsole

Erstellen eines Klons mithilfe der AWS-Managementkonsole Ergebnisse in einem Amazon DocumentDB-Cluster mit einer Amazon DocumentDB-Instance.

Diese Anweisungen gelten für DB-Cluster, die demselben AWS Konto gehören, das den Clone erstellt. Der DB-Cluster muss demselben AWS Konto gehören, da kontoübergreifendes Klonen in Amazon DocumentDB nicht unterstützt wird.

Um einen Klon eines DB-Clusters zu erstellen, das Ihrem gehört AWS Konto mit dem AWS-Managementkonsole

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Klicken Sie im Navigationsbereich auf Cluster.
3. Wählen Sie Ihren Amazon DocumentDB-Cluster aus der Liste aus und wählen Sie für Aktionen die Option Create clone aus.

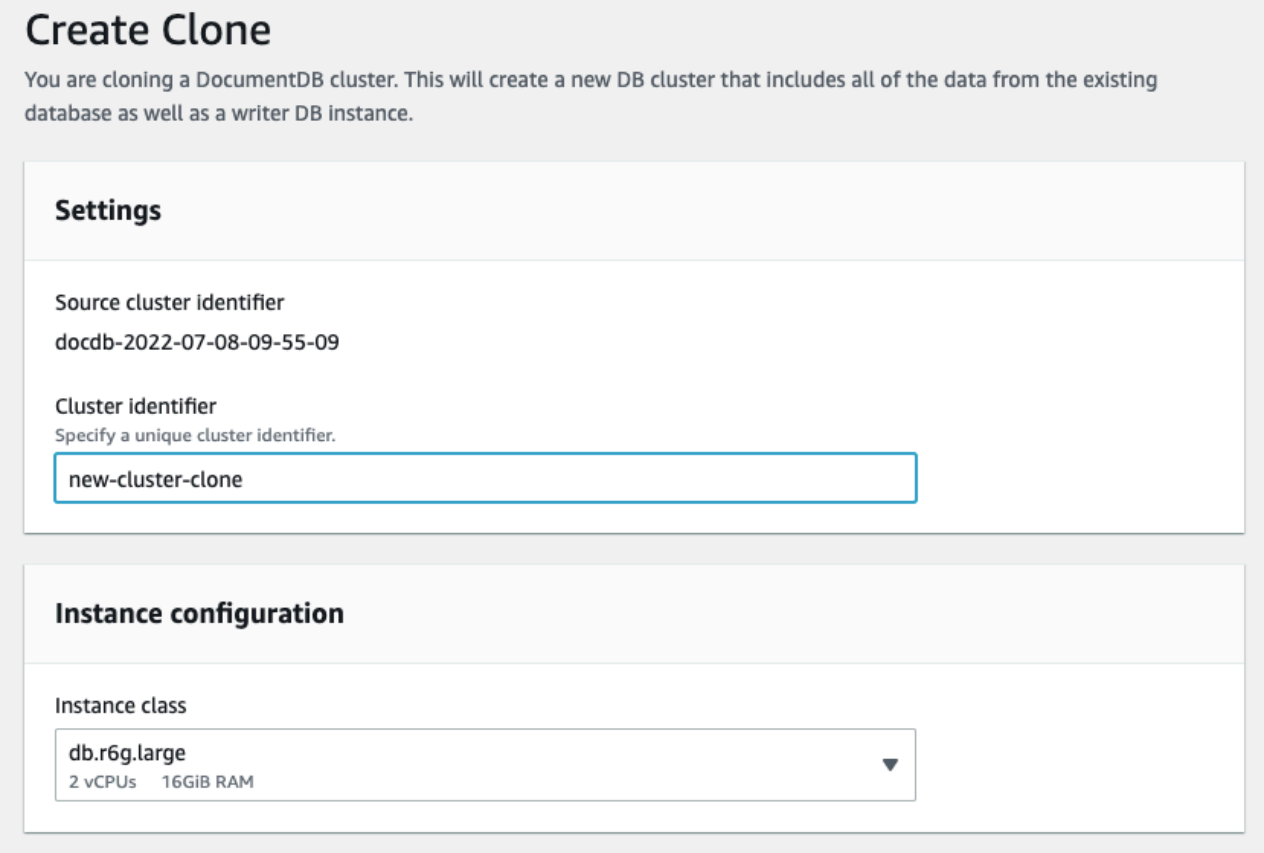


Die Seite „Clone erstellen“ wird geöffnet, auf der Sie eine Cluster-ID und eine Instance-Klasse sowie andere Optionen für den Amazon DocumentDB-Cluster-Clon konfigurieren können.

4. Gehen Sie im Abschnitt Settings (Einstellungen) wie folgt vor:

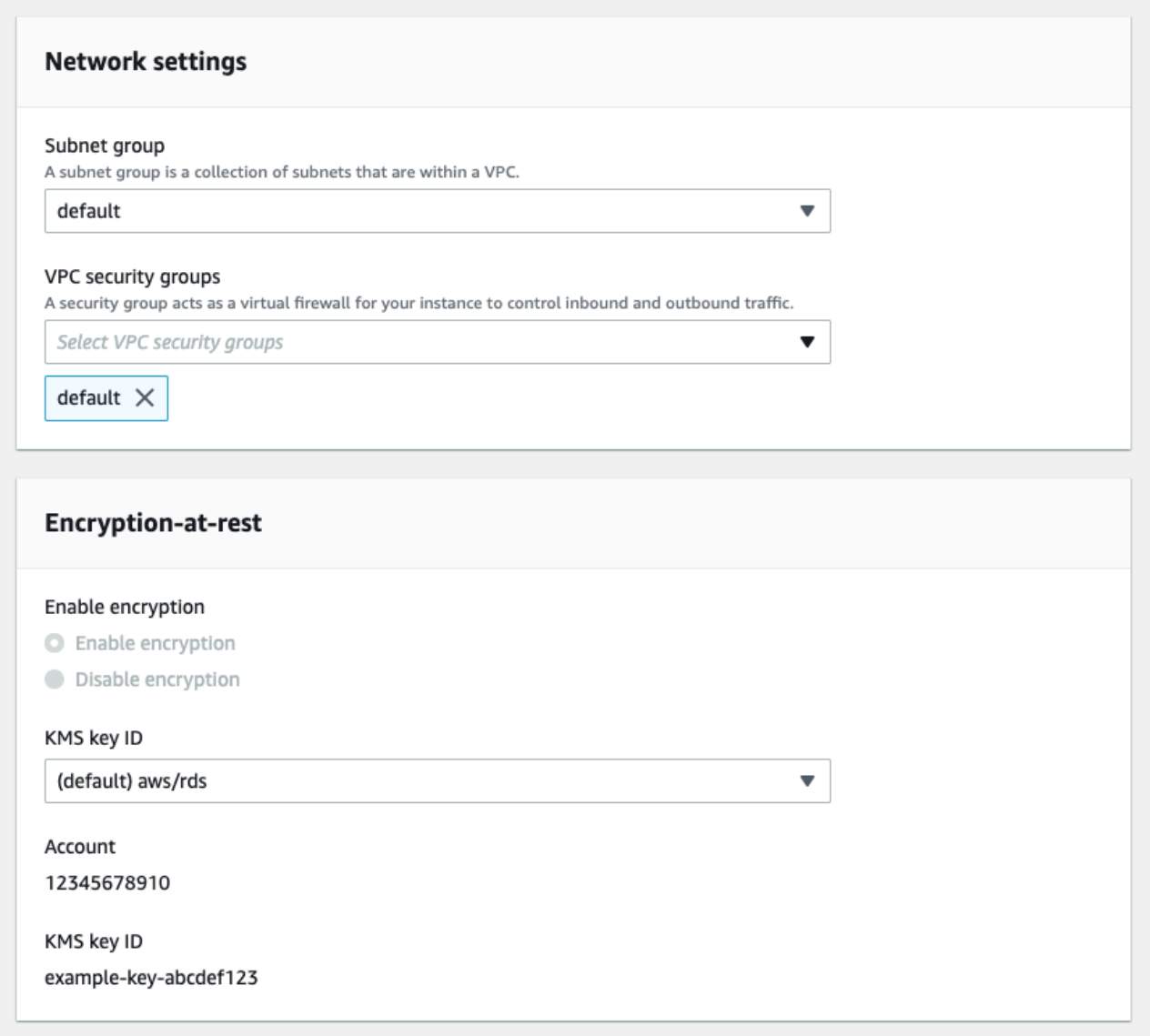
a. Geben Sie als Cluster-ID den Namen ein, den Sie Ihrem geklonten Amazon DocumentDB-Cluster geben möchten.

b. Wählen Sie für die Instance-Konfiguration eine geeignete Instance-Klasse für Ihren geklonten Amazon DocumentDB-Cluster aus.

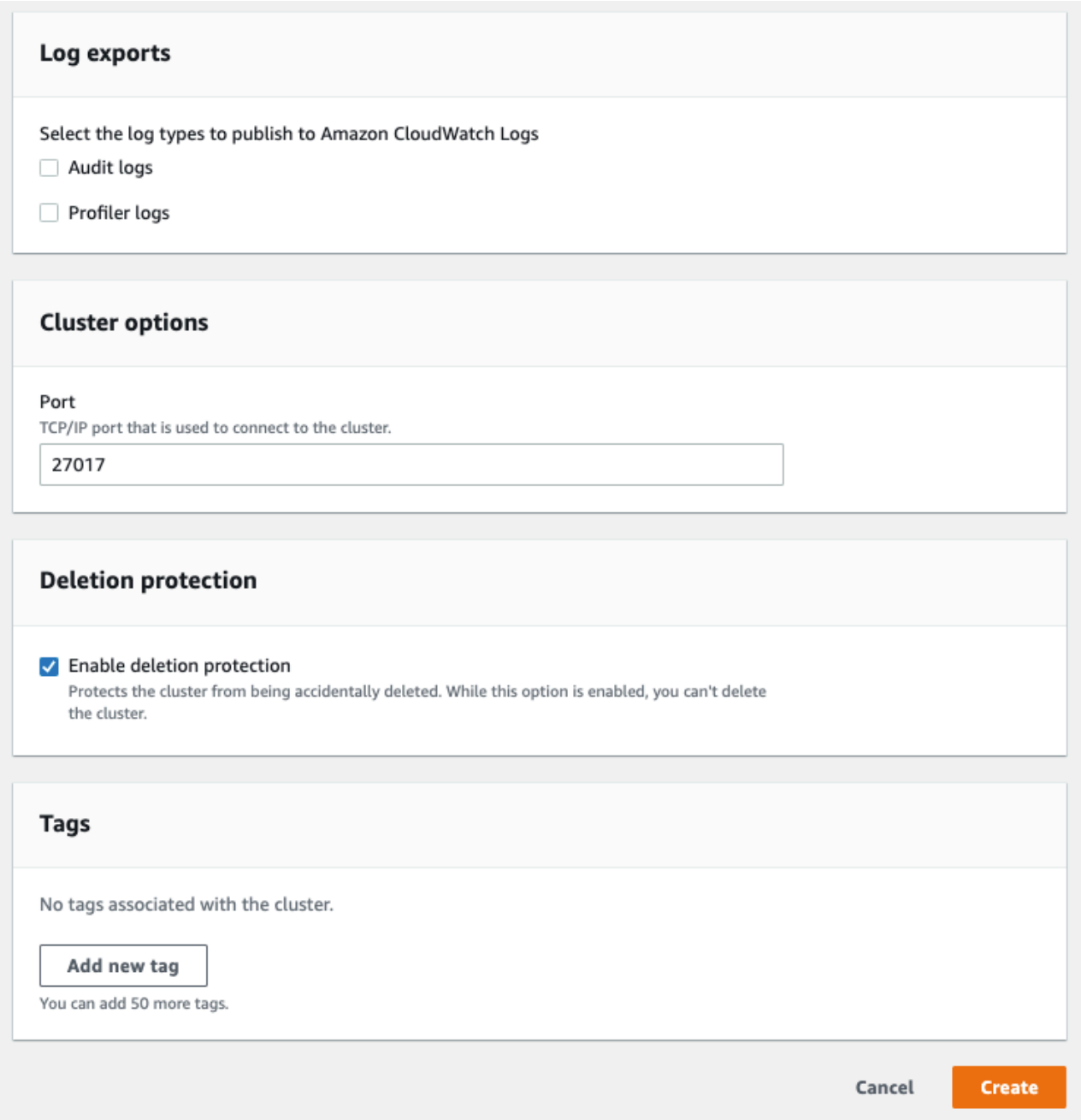


c. Wählen Sie für Netzwerkeinstellungen eine Subnetzgruppe für Ihren Anwendungsfall und die zugehörigen VPC-Sicherheitsgruppen aus.

d. Denn wenn für den Quellcluster (der Cluster Encryption-at-rest, der geklont wird) die Verschlüsselung aktiviert ist, muss auch für den geklonten Cluster die Verschlüsselung aktiviert sein. Wenn dieses Szenario zutrifft, sind die Optionen Verschlüsselung aktivieren ausgegraut (deaktiviert), aber die Option Verschlüsselung aktivieren ist ausgewählt. Umgekehrt sind die Optionen Verschlüsselung aktivieren verfügbar, wenn für den Quellcluster keine Verschlüsselung aktiviert ist, und Sie können wählen, ob die Verschlüsselung aktiviert oder deaktiviert werden soll.



e. Vervollständigen Sie die neue Cluster-Clone-Konfiguration, indem Sie den Typ der zu exportierenden Protokolle auswählen (optional), einen bestimmten Port eingeben, der für die Verbindung zum Cluster verwendet wird, und den Schutz vor versehentlichem Löschen des Clusters aktivieren (standardmäßig aktiviert).



f. Schließen Sie die Eingabe aller Einstellungen für Ihren Amazon DocumentDB-Cluster-Clon ab. Weitere Informationen zu den Amazon DocumentDB-Cluster- und Instance-Einstellungen finden Sie unter. Verwaltung von Amazon DocumentDB-Clustern

5. Wählen Sie Create Clone, um den Amazon DocumentDB-Clon Ihres ausgewählten Amazon DocumentDB-Clusters zu starten.

Wenn der Clone erstellt wurde, wird er zusammen mit Ihren anderen Amazon DocumentDB-Clustern im Abschnitt Datenbanken der Konsole aufgeführt und zeigt seinen aktuellen Status an. Ihr Klon ist einsatzbereit, wenn sein Status Verfügbar ist.

Using the AWS CLI

Die Verwendung des AWS CLI zum Klonen Ihres Amazon DocumentDB-Clusters umfasst einige Schritte.

Der `restore-db-cluster-to-point-in-time` AWS CLI Befehl, den Sie verwenden, führt zu einem leeren Amazon DocumentDB-Cluster mit 0 Amazon DocumentDB-Instances. Das heißt, der Befehl stellt nur den Amazon DocumentDB-Cluster wieder her, nicht die DB-Instances für diesen Cluster. Sie tun dies separat, nachdem der Klon verfügbar ist. Die zwei Schritte im Prozess sind wie folgt:

1. Erstellen Sie den Klon mit dem CLI-Befehl restore-db-cluster-to-point-in-time. Die Parameter, die Sie mit diesem Befehl verwenden, steuern den Kapazitätstyp und andere Details des leeren Amazon DocumentDB-Clusters (Clone), der gerade erstellt wird.
2. Erstellen Sie die Amazon DocumentDB-Instance für den Clone, indem Sie den CLI-Befehl create-db-instance verwenden, um die Amazon DocumentDB-Instance im wiederhergestellten Amazon DocumentDB-Cluster neu zu erstellen.

Bei den folgenden Befehlen wird davon ausgegangen, dass Ihre Region als AWS CLI Standard eingerichtet ist. AWS Dieser Ansatz erspart Ihnen die Übergabe des `--region`-Namens in jedem der Befehle. Weitere Informationen finden Sie unter Konfigurieren der AWS CLI. Sie können die `--region` auch in jedem der folgenden CLI-Befehle angeben.

Den Klon erstellen

Die spezifischen Parameter, die Sie an den `restore-db-cluster-to-point-in-time`-CLI-Befehl übergeben, variieren. Was Sie weitergeben, hängt von der Art des Klons ab, den Sie erstellen möchten.

Verwenden Sie das folgende Verfahren, um einen bereitgestellten Amazon DocumentDB-Clon aus einem bereitgestellten Amazon DocumentDB-Cluster zu erstellen.

Um einen Klon desselben Engine-Modus wie der Amazon DocumentDB-Quell-Cluster zu erstellen

- Verwenden Sie den `restore-db-cluster-to-point-in-time`-CLI-Befehl und geben Sie Werte für die folgenden Parameter an:

- `--db-cluster-identifier` – Wählen Sie einen aussagekräftigen Namen für Ihren Klon. Sie benennen den Klon, wenn Sie den CLI-Befehl restore-db-cluster-to-point-in-time verwenden.
- `--restore-type` – Verwenden Sie `copy-on-write`, um einen Klon des Quell-DB-Clusters zu erstellen. Ohne diesen Parameter `restore-db-cluster-to-point-in-time` stellt der den Amazon DocumentDB-Cluster wieder her, anstatt einen Klon zu erstellen. Die Standardeinstellung für `restore-type` ist`full-copy`.
- `--source-db-cluster-identifier`— Verwenden Sie den Namen des Amazon DocumentDB-Quell-Clusters, den Sie klonen möchten.
- `--use-latest-restorable-time` – Dieser Wert verweist auf die neuesten wiederherstellbaren Volume-Daten für den Klon. Dieser Parameter ist erforderlich`restore-type copy-on-write`, Sie können ihn jedoch nicht `restore-to-time parameter` damit verwenden.

Im folgenden Beispiel wird ein Klon namens `my-clone` aus einem Cluster namens `my-source-cluster` erstellt.

Für Linux, macOS oder Unix:

```
aws docdb restore-db-cluster-to-point-in-time \
    --source-db-cluster-identifier my-source-cluster \
    --db-cluster-identifier my-clone \
    --restore-type copy-on-write \
    --use-latest-restorable-time
```

Für Windows:

```
aws docdb restore-db-cluster-to-point-in-time ^
    --source-db-cluster-identifier my-source-cluster ^
    --db-cluster-identifier my-clone ^
    --restore-type copy-on-write ^
    --use-latest-restorable-time
```

Der Befehl gibt das JSON-Objekt zurück, das Details des Klons enthält. Stellen Sie sicher, dass Ihr geklonter DB-Cluster verfügbar ist, bevor Sie versuchen, die DB-Instance für Ihren Klon zu erstellen. Weitere Informationen finden Sie unten unter Status überprüfen und Clone-Details abrufen:

Überprüfen Sie den Status und rufen Sie die Clone-Details ab

Mit dem folgenden Befehl können Sie den Status Ihres neu erstellten leeren DB-Clusters überprüfen.

```
$ aws docdb describe-db-clusters --db-cluster-identifier my-clone --query '*[].
[Status]' --output text
```

Oder Sie können den Status und die anderen Werte, die Sie benötigen, um die DB-Instance für Ihren Clone zu erstellen, mithilfe der folgenden AWS CLI Abfrage abrufen:

Für Linux, macOS oder Unix:

```
aws docdb describe-db-clusters --db-cluster-identifier my-clone \
  --query '*[].{Status:Status,Engine:Engine,EngineVersion:EngineVersion}'
```

Für Windows:

```
aws docdb describe-db-clusters --db-cluster-identifier my-clone ^
  --query "*[].{Status:Status,Engine:Engine,EngineVersion:EngineVersion}"
```

Diese Abfrage gibt eine Ausgabe ähnlich der folgenden zurück.

```
[
  {
        "Status": "available",
        "Engine": "docdb",
        "EngineVersion": "4.0.0",
    }
]
```

Erstellen Sie die Amazon DocumentDB-Instance für Ihren Clone

Verwenden Sie den CLI-Befehl create-db-instance, um die DB-Instance für Ihren Klon zu erstellen.

Der `--db-instance-class` Parameter wird nur für bereitgestellte Amazon DocumentDB-Cluster verwendet.

Für Linux, macOS oder Unix:

```
aws docdb create-db-instance \
    --db-instance-identifier my-new-db \
    --db-cluster-identifier my-clone \
    --db-instance-class  db.r5.4xlarge \
    --engine docdb
```

Für Windows:

```
aws docdb create-db-instance ^
    --db-instance-identifier my-new-db ^
    --db-cluster-identifier my-clone ^
    --db-instance-class  db.r5.4xlarge ^
    --engine docdb
```

Parameter, die für das Klonen verwendet werden sollen

In der folgenden Tabelle sind die verschiedenen Parameter zusammengefasst, die beim Klonen von Amazon `restore-db-cluster-to-point-in-time` DocumentDB-Clustern verwendet werden.

Parameter	Beschreibung
--source-db-cluster-identifier	Verwenden Sie den Namen des Amazon DocumentDB-Quell-Clusters, den Sie klonen möchten.
--db-cluster-identifier	Wählen Sie einen aussagekräftigen Namen für Ihren Klon. Sie benennen Ihren Klon mit dem `restore-db-cluster-to-point-in-time` -Befehl. Dann übergeben Sie diesen Namen an den `create-db-instance` -Befehl.
--restore-type	Geben Sie `copy-on-write` als an`--restore-type` , um einen Klon des Quell-DB-Clusters zu erstellen, anstatt den Amazon DocumentDB-Quellcluster wiederherzustellen.
--use-latest-restorable-time	ieser Wert verweist auf die neuesten wiederherstellbaren Volume-Daten für den Klon.

Grundlegendes zur Amazon DocumentDB-Cluster-Fehlertoleranz

Amazon DocumentDB-Cluster sind von Haus aus fehlertolerant. Jedes Cluster-Volume erstreckt sich über mehrere Availability Zones in einer einzigen AWS-Region, und jede Availability Zone enthält eine Kopie der Volume-Daten des Clusters. Diese Funktionalität bedeutet, dass Ihr Cluster einen Ausfall der Availability Zone ohne Datenverlust und nur eine kurze Unterbrechung des Services tolerieren kann.

Wenn die primäre Instance in einem Cluster ausfällt, führt Amazon DocumentDB automatisch ein Failover zu einer neuen primären Instance durch, und zwar auf eine von zwei Arten:

- Durch Heraufstufen einer vorhandenen Amazon DocumentDB-Replica auf die neue primäre Instance, die auf der Grundlage der Promotion-Stufeneinstellung der einzelnen Replikate ausgewählt wird, und dann wird eine Ersatzinstanz für die vorherige primäre Instance erstellt. Ein Failover zur Replikat-Instance dauert in der Regel weniger als 30 Sekunden. Während dieser Zeit kann es bei Lese- und Schreibvorgängen zu kurzen Unterbrechungen kommen. Um die Verfügbarkeit Ihres Clusters zu erhöhen, erstellen Sie mindestens eine oder mehrere Amazon DocumentDB-Repliken in zwei oder mehr verschiedenen Availability Zones.
- Über das Anlegen einer neuen primären Instance. Dies passiert nur, wenn Sie keine Replikat-Instance in Ihrem Cluster haben und es kann einige Minuten dauern, bis der Vorgang abgeschlossen ist.

Wenn der Cluster über eine oder mehrere Amazon DocumentDB-Repliken verfügt, wird bei einem Ausfall eine Amazon DocumentDB-Replica zur primären Instance hochgestuft. Ein Fehlerereignis hat eine kurze Unterbrechung zufolge, während die Lese- und Schreibvorgänge mit einer Ausnahme fehlschlagen. Jedoch wird der Service im Normalfall in weniger als 120 Sekunden und oft sogar schon nach 60 Sekunden wiederhergestellt. Um die Verfügbarkeit Ihres Clusters zu erhöhen, erstellen Sie mindestens eine oder mehrere Amazon DocumentDB-Repliken in zwei oder mehr verschiedenen Availability Zones.

Sie können die Reihenfolge anpassen, in der Ihre Amazon DocumentDB-Replikate nach einem Ausfall zur primären Instance hochgestuft werden, indem Sie jedem Replikat eine Priorität zuweisen. Prioritäten liegen im Bereich zwischen 0 als höchste Priorität und 15 als niedrigste Priorität. Wenn die primäre Instance ausfällt, wird das Amazon DocumentDB-Replikat mit der höchsten Priorität auf die neue primäre Instance heraufgestuft. Sie können die Priorität einer Amazon DocumentDB-Replica jederzeit ändern. Das Ändern der Priorität löst kein Failover aus. Sie können die `modify-db-`

`instance`-Operation mit dem `--promotion-tier`-Parameter verwenden. Weitere Informationen zum Anpassen der Failover-Priorität einer Instance finden Sie unter Amazon DocumentDB-Failover.

Mehr als ein Amazon DocumentDB-Replikat kann dieselbe Priorität haben, was zu Werbestufen führt. Wenn zwei oder mehr Amazon DocumentDB-Replikate dieselbe Priorität haben, wird das Replikat mit der größten Größe auf das primäre Replikat heraufgestuft. Wenn zwei oder mehr Amazon DocumentDB-Repliken dieselbe Priorität und Größe haben, wird ein beliebiges Replikat derselben Werbestufe hochgestuft.

Wenn der Cluster keine Amazon DocumentDB-Replikate enthält, wird die primäre Instance bei einem Ausfall neu erstellt. Ein Fehlerereignis hat eine Unterbrechung zufolge, während die Lese- und Schreibvorgänge mit einer Ausnahme fehlschlagen. Der Service wird wiederhergestellt, wenn die primäre Instance erstellt wird. Dies dauert im Normalfall weniger als 10 Minuten. Das Heraufstufen einer Amazon DocumentDB-Replica zur primären Instance ist viel schneller als das Erstellen einer neuen primären Instance.

Verwaltung von Amazon DocumentDB DocumentDB-Instances

Die folgenden Themen enthalten Informationen, die Ihnen bei der Verwaltung Ihrer Amazon DocumentDB DocumentDB-Instances helfen sollen. Sie umfassen Details über Instance-Klassen und Statusarten und das Erstellen, Löschen und Ändern einer Instance.

Themen

- Den Status einer Instance ermitteln
- Lebenszyklus einer Amazon DocumentDB DocumentDB-Instanz
- Instanzklassen verwalten
- NVMe-backed Instanzen

Den Status einer Instance ermitteln

Mehr zu den gültigen Instance-Zuständen, ihre Bedeutung und die Bestimmung des Status Ihrer Instances finden Sie unter Überwachung des Status einer Amazon DocumentDB-Instance.

Lebenszyklus einer Amazon DocumentDB DocumentDB-Instanz

Der Lebenszyklus einer Amazon DocumentDB DocumentDB-Instance umfasst das Erstellen, Ändern, Verwalten und Aktualisieren, das Durchführen von Backups und Wiederherstellungen, das

Neustarten und Löschen der Instance. Dieser Abschnitt enthält Informationen darüber, wie Sie diese Prozesse abschließen können.

Themen

- Hinzufügen einer Amazon DocumentDB DocumentDB-Instance zu einem Cluster
- Beschreibung von Amazon DocumentDB DocumentDB-Instances
- Ändern einer Amazon DocumentDB DocumentDB-Instance
- Eine Amazon DocumentDB DocumentDB-Instance neu starten
- Löschen einer Amazon DocumentDB DocumentDB-Instance

Sie können eine neue Amazon DocumentDB DocumentDB-Instance mit dem AWS-Managementkonsole oder dem AWS CLI erstellen. Um eine Instance zu einem Cluster hinzuzufügen, muss sich der Cluster im Zustand available befinden. Zu einem angehaltenen Cluster können keine Instances hinzugefügt werden. Wenn der Cluster angehalten ist, starten Sie zuerst den Cluster, warten Sie, bis der Cluster verfügbar wird, und fügen Sie dann eine Instance hinzu. Weitere Informationen finden Sie unter Stoppen und Starten eines Amazon DocumentDB-Clusters.

> **Note**
>
> Wenn Sie mit der Konsole einen Amazon DocumentDB-Cluster erstellen, wird gleichzeitig automatisch eine Instance für Sie erstellt. Wenn Sie zusätzliche Instances erstellen möchten, verwenden Sie eines der folgenden Verfahren.

Hinzufügen einer Amazon DocumentDB DocumentDB-Instance zu einem Cluster

Using the AWS-Managementkonsole

Gehen Sie wie folgt vor, um mithilfe der Amazon DocumentDB DocumentDB-Konsole eine Instance für Ihren Cluster zu erstellen.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Klicken Sie im Navigationsbereich auf Cluster.

> **Tip**
>
> Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Um den Cluster zu wählen, dem Sie eine Instance hinzufügen möchten, wählen Sie die Schaltfläche links vom Namen des Clusters aus.
4. Wählen Sie Actions (Aktionen) und dann Add instances (Instances hinzufügen) aus.
5. Wiederholen Sie die folgenden Schritte für jede Instance, die Sie dem Cluster hinzufügen möchten, auf der Seite Add instance to: (Instance hinzufügen zu:)<cluster-name>. Sie können bis zu 15 haben.

   a. Instance-ID — Sie können entweder eine eindeutige Kennung für diese Instance eingeben oder Amazon DocumentDB erlauben, die Instance-ID auf der Grundlage der Cluster-ID bereitzustellen.

      Einschränkungen für Instance-Benennungen:

      - Die Länge beträgt [1—63] Buchstaben, Zahlen oder Bindestriche.
      - Muss mit einem Buchstaben beginnen.
      - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.
      - Muss für alle Instances in Amazon RDS, Neptune und Amazon DocumentDB pro AWS-Konto Region eindeutig sein.

   b. Instance-Klasse — Wählen Sie aus der Drop-down-Liste den Instance-Typ aus, den Sie für diese Instance benötigen.

   c. Promotion-Stufe — Wählen Sie aus der Drop-down-Liste die Promotion-Stufe für Ihre Instance oder wählen Sie Keine Präferenz, damit Amazon DocumentDB die Promotion-Stufe für Ihre Instance festlegen kann. Niedrigere Nummern bedeuten eine höhere Priorität. Weitere Informationen finden Sie unter Steuerung des Failover-Ziels.

   d. Zum Hinzufügen weiterer Instances wählen Sie die Option Add additional instances (Weitere Instances hinzufügen) und wiederholen Sie die Schritte a, b und c.

6. Schließen Sie den Vorgang ab.

- Wenn Sie die Instances zu Ihrem Cluster hinzufügen möchten, klicken Sie auf Create (Erstellen).
- Um die Operation abzubrechen, wählen Sie Abbrechen aus.

Es dauert einige Minuten, bis eine Instance erstellt ist. Sie können die Konsole verwenden oder AWS CLI den Status der Instance einsehen. Weitere Informationen finden Sie unter Überwachung des Status einer Instanz.

Using the AWS CLI

Verwenden Sie den `create-db-instance` AWS CLI Vorgang mit den folgenden Parametern, um die primäre Instance für Ihren Cluster zu erstellen.

- **`--db-instance-class`** – Erforderlich. Die Rechen- und Speicherkapazität der Instance, beispielsweise `db.m4.large`. Nicht alle Instanzklassen sind in allen verfügbar AWS-Regionen.
- **`--db-instance-identifier`** – Erforderlich. Eine -Zeichenfolge in Kleinbuchstaben, die die Instance bezeichnet.

  Instance-Benennungseinschränkungen:

  - Die Länge beträgt [1—63] Buchstaben, Zahlen oder Bindestriche.
  - Muss mit einem Buchstaben beginnen.
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.
  - Muss für alle Instances in Amazon RDS, Neptune und Amazon DocumentDB pro AWS-Konto Region eindeutig sein.
- **`--engine`** – Erforderlich. Der Wert muss `docdb` sein.
- **`--availability-zone`**— Fakultativ. Die Availability Zone, in der diese Instance erstellt werden soll. Verwenden Sie diesen Parameter, um Ihre Instances in verschiedenen Availability Zones zur Erhöhung der Fehlertoleranz zu finden. Weitere Informationen finden Sie unter Amazon DocumentDB — Hochverfügbarkeit und Replikation.
- **`--promotion-tier`**— Fakultativ. Die Failover-Prioritätsstufe für diese Instance. Muss zwischen 0 und 15 liegen, wobei niedrigere Nummern eine höhere Priorität bedeuten. Weitere Informationen finden Sie unter Steuerung des Failover-Ziels.

1. Bestimmen Sie zunächst, in welchen Availability Zones Sie Ihre Instance erstellen können.

Wenn Sie die Availability Zone angeben möchten, bevor Sie Ihre Instance erstellen, führen Sie den folgenden Befehl aus, um zu ermitteln, welche Availability Zones für Ihren Amazon DocumentDB-Cluster verfügbar sind.

Für Linux, macOS oder Unix:

```
aws docdb describe-db-clusters \
       --query 'DBClusters[*].[DBClusterIdentifier,AvailabilityZones[*]]'
```

Für Windows:

```
aws docdb describe-db-clusters ^
       --query 'DBClusters[*].[DBClusterIdentifier,AvailabilityZones[*]]'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
[
    [
        "sample-cluster",
        [
            "us-east-1c",
            "us-east-1b",
            "us-east-1a"
        ]
    ]
]
```

2. Bestimmen Sie dann, welche Instance-Klassen Sie in Ihrer Region erstellen können.

   Führen Sie den folgenden Befehl aus, um festzustellen, welche Instance-Klassen Ihnen in Ihrer Region zur Verfügung stehen. Wählen Sie aus der Ausgabe eine Instance-Klasse für die Instance aus, die Sie Ihrem Amazon DocumentDB-Cluster hinzufügen möchten.

   Für Linux, macOS oder Unix:

   ```
   aws docdb describe-orderable-db-instance-options \
           --engine docdb \
           --query 'OrderableDBInstanceOptions[*].DBInstanceClass'
   ```

Für Windows:

```
aws docdb describe-orderable-db-instance-options ^
        --engine docdb ^
        --query 'OrderableDBInstanceOptions[*].DBInstanceClass'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
[
    "db.r5.16xlarge",
    "db.r5.2xlarge",
    "db.r5.4xlarge",
    "db.r5.8xlarge",
    "db.r5.large",
    "db.r5.xlarge"
]
```

3. Fügen Sie abschließend eine Instance zu Ihrem Amazon DocumentDB-Cluster hinzu.

   Um Ihrem Amazon DocumentDB-Cluster eine Instance hinzuzufügen, führen Sie den folgenden Befehl aus.

   Für Linux, macOS oder Unix:

```
aws docdb create-db-instance \
       --db-cluster-identifier sample-cluster \
       --db-instance-identifier sample-instance-2 \
       --availability-zone us-east-1b \
       --promotion-tier 2 \
       --db-instance-class db.r5.xlarge \
       --engine docdb
```

   Für Windows:

```
aws docdb create-db-instance ^
       --db-cluster-identifier sample-cluster ^
       --db-instance-identifier sample-instance-2 ^
       --availability-zone us-east-1b ^
       --promotion-tier 2 ^
       --db-instance-class db.r5.xlarge ^
```

```
        --engine docdb
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "sample-instance-2",
        "DBInstanceClass": "db.r5.xlarge",
        "Engine": "docdb",
        "DBInstanceStatus": "creating",
        "PreferredBackupWindow": "02:00-02:30",
        "BackupRetentionPeriod": 1,
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abcd0123",
                "Status": "active"
            }
        ],
        "AvailabilityZone": "us-east-1b",
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-6242c31a",
            "SubnetGroupStatus": "Complete",
            "Subnets": [
                {
                    "SubnetIdentifier": "subnet-abcd0123",
                    "SubnetAvailabilityZone": {
                        "Name": "us-west-2a"
                    },
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-wxyz0123",
                    "SubnetAvailabilityZone": {
                        "Name": "us-west-2b"
                    },
                    "SubnetStatus": "Active"
                }
            ]
        },
        "PreferredMaintenanceWindow": "sun:11:35-sun:12:05",
        "PendingModifiedValues": {},
```

```
        "EngineVersion": "3.6.0",
        "AutoMinorVersionUpgrade": true,
        "PubliclyAccessible": false,
        "DBClusterIdentifier": "sample-cluster",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/sample-key",
        "DbiResourceId": "db-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
        "CACertificateIdentifier": "rds-ca-2019",
        "PromotionTier": 2,
        "DBInstanceArn": "arn:aws:rds:us-east-1:<accountID>:db:sample-instance-2"
    }
}
```

Es dauert einige Minuten, bis die Instance erstellt ist. Sie können die Konsole verwenden oder AWS CLI den Status der Instance einsehen. Weitere Informationen finden Sie unter Überwachung des Status einer Amazon DocumentDB-Instance.

Beschreibung von Amazon DocumentDB DocumentDB-Instances

Sie können entweder die Amazon DocumentDB-Managementkonsole oder die verwenden, AWS CLI um Details wie Verbindungsendpunkte, Sicherheitsgruppen, VPCs, Zertifizierungsstellen und Parametergruppen zu Ihren Amazon DocumentDB DocumentDB-Instances anzuzeigen.

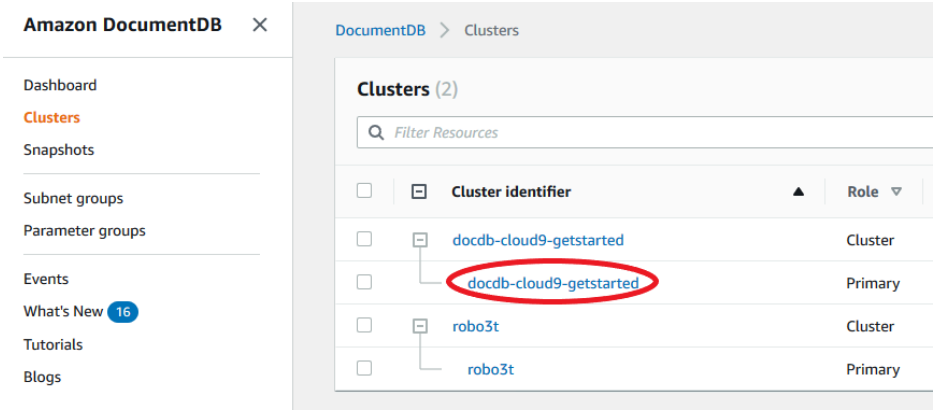
Using the AWS-Managementkonsole

Gehen Sie folgendermaßen vor, um die Details Ihrer Instances mithilfe der AWS-Managementkonsole anzuzeigen.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Klicken Sie im Navigationsbereich auf Cluster.

> **Tip**
>
> Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Im Cluster-Navigationsfeld sehen Sie die Spalte Cluster-ID. Ihre Instances werden unter Clustern aufgeführt, ähnlich wie im folgenden Screenshot.



4. Wählen Sie in der Liste der Instances den Namen der Instanz aus, deren Details Sie sehen möchten. Die Informationen über die Instance sind in die folgenden Gruppierungen unterteilt:

   - Zusammenfassung — Allgemeine Informationen über die Instanz, einschließlich der Engine-Version, der Klasse, des Status und aller ausstehenden Wartungsarbeiten.
   - Konnektivität und Sicherheit — Der Abschnitt Connect listet die Verbindungsendpunkte auf, um mit der Mongo-Shell oder mit einer Anwendung eine Verbindung zu dieser Instanz herzustellen. Im Abschnitt Sicherheitsgruppen werden die Sicherheitsgruppen aufgeführt, die dieser Instance zugeordnet sind, sowie ihre VPC-ID und Beschreibungen.
   - Konfiguration — Der Abschnitt „Details“ listet die Konfigurationen und den Status der Instance auf, einschließlich des Amazon-Ressourcennamens (ARN), des Endpunkts, der Rolle, der Klasse und der Zertifizierungsstelle der Instance. Außerdem werden die Sicherheits- und Netzwerkeinstellungen der Instance sowie die Sicherungsinformationen aufgelistet. Im Abschnitt Cluster-Details werden die Details des Clusters aufgeführt, zu dem diese Instance gehört. Im Abschnitt Cluster-Instances werden alle Instances aufgeführt, die zu Ihrem Cluster gehören, mit dem Status der einzelnen Instance-Rollen und Cluster-Parametergruppen.

     > **Note**
     >
     > Sie können den mit Ihrer Instance verknüpften Cluster ändern, indem Sie neben der Überschrift Cluster-Details auf Ändern klicken. Weitere Informationen finden Sie unter Ändern eines Amazon DocumentDB-Clusters.

   - Überwachung — Die CloudWatch Log-Metriken für diese Instance. Weitere Informationen finden Sie unter Überwachung von Amazon DocumentDB mit CloudWatch.

- Ereignisse und Tags — Der Abschnitt Aktuelle Ereignisse listet die jüngsten Ereignisse für diese Instance auf. Amazon DocumentDB zeichnet Ereignisse auf, die sich auf Ihre Cluster, Instances, Snapshots, Sicherheitsgruppen und Cluster-Parametergruppen beziehen. Zu diesen Informationen gehören Datum, Uhrzeit und Nachricht, die jedem Ereignis zugeordnet sind. Der Abschnitt Tags listet die an diesen Cluster angehängten Tags auf. Weitere Informationen finden Sie unter Taggen von Amazon DocumentDB DocumentDB-Ressourcen.

Using the AWS CLI

Um die Details Ihrer Amazon DocumentDB DocumentDB-Instances mithilfe von anzuzeigen AWS CLI, verwenden Sie den `describe-db-clusters` Befehl, wie in den folgenden Beispielen gezeigt. Weitere Informationen finden Sie `DescribeDBInstances`in der Amazon DocumentDB Resource Management API-Referenz.

> **Note**
>
> Für bestimmte Verwaltungsfunktionen wie Cluster- und Instance-Lebenszyklusmanagement nutzt Amazon DocumentDB Betriebstechnologie, die mit Amazon RDS gemeinsam genutzt wird. Der `filterName=engine,Values=docdb` Filterparameter gibt nur Amazon DocumentDB-Cluster zurück.

1. Listet alle Amazon DocumentDB DocumentDB-Instances auf.

   Der folgende AWS CLI Code listet die Details für alle Amazon DocumentDB DocumentDB-Instances in einer Region auf.

   Für Linux, macOS oder Unix:

   ```
   aws docdb describe-db-instances \
       --filter Name=engine,Values=docdb
   ```

   Für Windows:

   ```
   aws docdb describe-db-instances \
       --filter Name=engine,Values=docdb
   ```

2. Alle Details für eine angegebene Amazon DocumentDB DocumentDB-Instance auflisten

   Der folgende Code listet die Details für `sample-cluster-instance` auf. Das Einschließen des Parameters `--db-instance-identifier` mit dem Namen einer Instance beschränkt die Ausgabe auf Informationen zu dieser bestimmten Instance.

   Für Linux, macOS oder Unix:

```
aws docdb describe-db-instances \
    --db-instance-identifier sample-cluster-instance
```

   Für Windows:

```
aws docdb describe-db-instances \
    --db-instance-identifier sample-cluster-instance
```

   Die Ausgabe dieser Operation sieht folgendermaßen aus.

```
{
    "DBInstances": [
        {
            "DbiResourceId": "db-BJKKB54PIDV5QFKGVRX5T3S6GM",
            "DBInstanceArn": "arn:aws:rds:us-east-1:012345678901:db:sample-
cluster-instance-00",
            "VpcSecurityGroups": [
                {
                    "VpcSecurityGroupId": "sg-77186e0d",
                    "Status": "active"
                }
            ],
            "DBInstanceClass": "db.r5.large",
            "DBInstanceStatus": "creating",
            "AutoMinorVersionUpgrade": true,
            "PreferredMaintenanceWindow": "fri:09:32-fri:10:02",
            "BackupRetentionPeriod": 1,
            "StorageEncrypted": true,
            "DBClusterIdentifier": "sample-cluster",
            "EngineVersion": "3.6.0",
            "AvailabilityZone": "us-east-1a",
            "Engine": "docdb",
            "PromotionTier": 2,
```

```
            "DBInstanceIdentifier": "sample-cluster-instance",
            "PreferredBackupWindow": "00:00-00:30",
            "PubliclyAccessible": false,
            "DBSubnetGroup": {
                "DBSubnetGroupName": "default",
                "Subnets": [
                    {
                        "SubnetIdentifier": "subnet-4e26d263",
                        "SubnetAvailabilityZone": {
                            "Name": "us-east-1a"
                        },
                        "SubnetStatus": "Active"
                    },
                    {
                        "SubnetIdentifier": "subnet-afc329f4",
                        "SubnetAvailabilityZone": {
                            "Name": "us-east-1c"
                        },
                        "SubnetStatus": "Active"
                    },
                    {
                        "SubnetIdentifier": "subnet-b3806e8f",
                        "SubnetAvailabilityZone": {
                            "Name": "us-east-1e"
                        },
                        "SubnetStatus": "Active"
                    },
                    {
                        "SubnetIdentifier": "subnet-53ab3636",
                        "SubnetAvailabilityZone": {
                            "Name": "us-east-1d"
                        },
                        "SubnetStatus": "Active"
                    },
                    {
                        "SubnetIdentifier": "subnet-991cb8d0",
                        "SubnetAvailabilityZone": {
                            "Name": "us-east-1b"
                        },
                        "SubnetStatus": "Active"
                    },
                    {
                        "SubnetIdentifier": "subnet-29ab1025",
                        "SubnetAvailabilityZone": {
```

```
                            "Name": "us-east-1f"
                        },
                        "SubnetStatus": "Active"
                    }
                ],
                "VpcId": "vpc-91280df6",
                "DBSubnetGroupDescription": "default",
                "SubnetGroupStatus": "Complete"
            },
            "PendingModifiedValues": {},
            "KmsKeyId": "arn:aws:kms:us-east-1:012345678901:key/0961325d-
a50b-44d4-b6a0-a177d5ff730b"
        }
    ]
}
```

Ändern einer Amazon DocumentDB DocumentDB-Instance

Sie können Ihre Amazon DocumentDB DocumentDB-Instance entweder mit dem AWS-Managementkonsole oder dem AWS CLIändern. Nur Instances im Status available können geändert werden. Eine angehaltene Instance kann nicht geändert werden. Wenn der Cluster angehalten ist, starten Sie zuerst den Cluster, warten Sie, bis die Instance verfügbar wird, und nehmen Sie dann die gewünschten Änderungen vor. Weitere Informationen finden Sie unter Stoppen und Starten eines Amazon DocumentDB-Clusters.

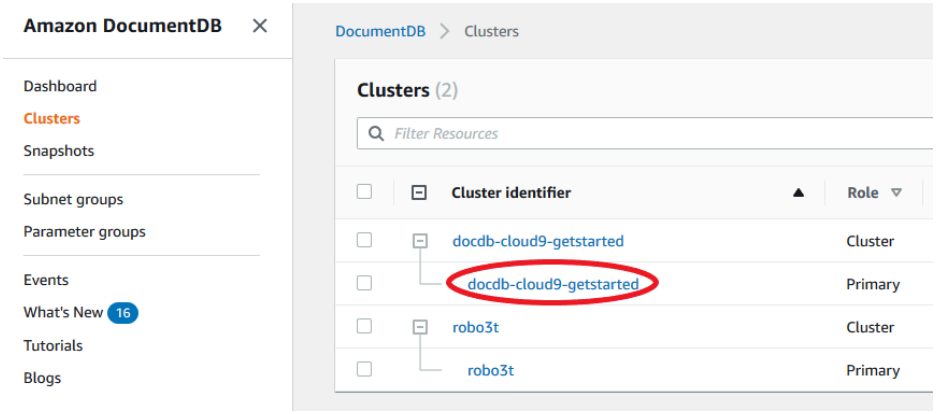
Using the AWS-Managementkonsole

Gehen Sie wie folgt vor, um eine bestimmte Amazon DocumentDB DocumentDB-Instance mithilfe der Konsole zu ändern.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Klicken Sie im Navigationsbereich auf Cluster.

> **Tip**
>
> Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Im Cluster-Navigationsfeld sehen Sie die Spalte Cluster-ID. Ihre Instances werden unter Clustern aufgeführt, ähnlich wie im folgenden Screenshot.



4. Markieren Sie das Kästchen links neben der Instanz, die Sie ändern möchten.
5. Wählen Sie Actions (Aktionen) und dann Modify (Ändern) aus.
6. Nehmen Sie im Bereich Modify instance: <instance-name> (Instance ändern: <instance-name>) die gewünschten Änderungen vor. Sie können die folgenden Änderungen ausführen:

   - Instanzspezifikationen — Die Instanz-ID und Klasse. Benennungseinschränkungen für Instance-IDs:
     - Instanz-ID — Geben Sie einen Namen ein, der für alle Instances, die Ihnen gehören, AWS-Konto in der aktuellen Region eindeutig ist. Die Instance-ID muss [1—63] alphanumerische Zeichen oder Bindestriche enthalten, einen Buchstaben als erstes Zeichen haben und darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.
     - Instance-Klasse — Wählen Sie im Drop-down-Menü eine Instance-Klasse für Ihre Amazon DocumentDB DocumentDB-Instance aus. Weitere Informationen finden Sie unter Instanzklassen verwalten.
   - Zertifizierungsstelle — Serverzertifikat für diese Instance. Weitere Informationen finden Sie unter Aktualisierung Ihrer Amazon DocumentDB-TLS-Zertifikate.

- Failover — Während des Failovers wird die Instanz mit der höchsten Heraufstufungsstufe zur primären Instanz heraufgestuft. Weitere Informationen finden Sie unter Amazon DocumentDB-Failover.
- Wartung — Das Wartungsfenster, in dem ausstehende Änderungen oder Patches auf Instanzen im Cluster angewendet werden.

7. Wenn Sie fertig sind, wählen Sie Continue (Weiter) aus, um eine Zusammenfassung Ihrer Änderungen anzuzeigen.
8. Nachdem Sie Ihre Änderungen überprüft haben, können Sie diese sofort oder während des nächsten Wartungsfensters unter Scheduling of modifications (Planen von Änderungen) anwenden. Klicken Sie auf Modify instance (Instance ändern), um Ihre Änderungen zu speichern. Alternativ können Sie auf Cancel (Abbrechen) klicken, um Ihre Änderungen zu verwerfen.

Es dauert einige Minuten, bis Ihre Änderungen übernommen werden. Sie können die Instance nur verwenden, wenn ihr Status available ist. Sie können mit der Konsole oder der AWS CLI den Status der Instance überwachen. Weitere Informationen finden Sie unter Überwachung des Status einer Amazon DocumentDB-Instance.

Using the AWS CLI

Um eine bestimmte Amazon DocumentDB DocumentDB-Instance mithilfe von zu ändern AWS CLI, verwenden Sie die `modify-db-instance` mit den folgenden Parametern. Weitere Informationen finden Sie unter ModifyDBInstance. Der folgende Code ändert die Instance-Klasse für die Instance `sample-instance` in `db.r5.large`.

Parameters

- **`--db-instance-identifier`** – Erforderlich. Die Kennung der zu ändernden Instance.
- **`--db-instance-class`**— Fakultativ. Die neue Rechen- und Speicherkapazität der Instanz; zum Beispiel`db.r5.large`. Nicht alle Instanzklassen sind in allen verfügbar AWS-Regionen. Wenn Sie die Instanzklasse ändern, kommt es während der Änderung zu einem Ausfall. Die Änderung wird im nächsten Wartungsfenster übernommen, sofern sie `ApplyImmediately` für diese Anfrage nicht als wahr angegeben wurde.
- **`--apply-immediately`**oder **`--no-apply-immediately`** — Optional. Gibt an, ob diese Änderung sofort angewandt oder bis zum nächsten Wartungszeitraum verschoben werden sollte. Wenn dieser Parameter weggelassen wird, wird die Änderung im nächsten Wartungszeitraum ausgeführt.

Example

Für Linux, macOS oder Unix:

```
aws docdb modify-db-instance \
       --db-instance-identifier sample-instance \
       --db-instance-class db.r5.large \
       --apply-immediately
```

Für Windows:

```
aws docdb modify-db-instance ^
       --db-instance-identifier sample-instance ^
       --db-instance-class db.r5.large ^
       --apply-immediately
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
    "DBInstances": [
        {
            "DBInstanceIdentifier": "sample-instance-1",
            "DBInstanceClass": "db.r5.large",
            "Engine": "docdb",
            "DBInstanceStatus": "modifying",
            "Endpoint": {
                "Address": "sample-instance-1.node.us-east-1.docdb.amazonaws.com",
                "Port": 27017,
                "HostedZoneId": "ABCDEFGHIJKLM"
            },
            "InstanceCreateTime": "2020-01-10T22:18:55.921Z",
            "PreferredBackupWindow": "02:00-02:30",
            "BackupRetentionPeriod": 1,
            "VpcSecurityGroups": [
                {
                    "VpcSecurityGroupId": "sg-abcd0123",
                    "Status": "active"
                }
            ],
            "AvailabilityZone": "us-east-1a",
            "DBSubnetGroup": {
                "DBSubnetGroupName": "default",
                "DBSubnetGroupDescription": "default",
```

```
                "VpcId": "vpc-abcd0123",
                "SubnetGroupStatus": "Complete",
                "Subnets": [
                    {
                        "SubnetIdentifier": "subnet-abcd0123",
                        "SubnetAvailabilityZone": {
                            "Name": "us-east-1a"
                        },
                        "SubnetStatus": "Active"
                    },
                    {
                        "SubnetIdentifier": "subnet-abcd0123",
                        "SubnetAvailabilityZone": {
                            "Name": "us-east-1b"
                        },
                        "SubnetStatus": "Active"
                    }
                ]
            },
            "PreferredMaintenanceWindow": "sun:10:57-sun:11:27",
            "PendingModifiedValues": {
                "DBInstanceClass": "db.r5.large"
            },
            "EngineVersion": "3.6.0",
            "AutoMinorVersionUpgrade": true,
            "PubliclyAccessible": false,
            "DBClusterIdentifier": "sample-cluster",
            "StorageEncrypted": true,
            "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/wJalrXUtnFEMI/
K7MDENG/bPxRfiCYEXAMPLEKEY",
            "DbiResourceId": "db-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
            "CACertificateIdentifier": "rds-ca-2019",
            "PromotionTier": 1,
            "DBInstanceArn": "arn:aws:rds:us-east-1:123456789012:db:sample-
instance-1",
            "EnabledCloudwatchLogsExports": [
                "profiler"
            ]
        }
    ]
}
```

Es dauert einige Minuten, bis Ihre Änderungen angewendet werden. Sie können die Instance nur verwenden, wenn ihr Status available ist. Sie können den Status der Instanz mit dem AWS-Managementkonsole oder überwachen AWS CLI. Weitere Informationen finden Sie unter Überwachung des Status einer Amazon DocumentDB-Instance.

Eine Amazon DocumentDB DocumentDB-Instance neu starten

Gelegentlich müssen Sie Ihre Amazon DocumentDB DocumentDB-Instance möglicherweise neu starten, normalerweise aus Wartungsgründen. Wenn Sie bestimmte Änderungen vornehmen, z. B. die Clusterparametergruppe, die einem Cluster zugeordnet ist, müssen Sie die Instances im Cluster neu starten, damit die Änderungen wirksam werden. Sie können eine angegebene Instance mit dem AWS-Managementkonsole oder dem AWS CLI neu starten.

Durch den Neustart einer Instance wird der Datenbank-Engine-Service neu gestartet. Das Neustarten bewirkt einen vorübergehenden Nutzungsausfall, wobei der Status für die Instance währenddessen auf `rebooting` gesetzt wird. Ein Amazon DocumentDB DocumentDB-Ereignis wird erstellt, wenn der Neustart abgeschlossen ist.

Ein Neustart einer Instance führt nicht zu einem Failover. Verwenden Sie die Operation AWS-Managementkonsole oder, um ein Failover für einen Amazon DocumentDB-Cluster durchzuführen. AWS CLI `failover-db-cluster` Weitere Informationen finden Sie unter Amazon DocumentDB-Failover.

Sie können Ihre Instance nicht neu starten, wenn sie sich nicht im verfügbaren Zustand befindet. Ihre Datenbank kann aus mehreren Gründen nicht verfügbar sein, zum Beispiel aufgrund einer zuvor angeforderten Änderung oder einer Aktion im Wartungsfenster. Weitere Informationen zu Instance-Status finden Sie unter Überwachung des Status einer Amazon DocumentDB-Instance.

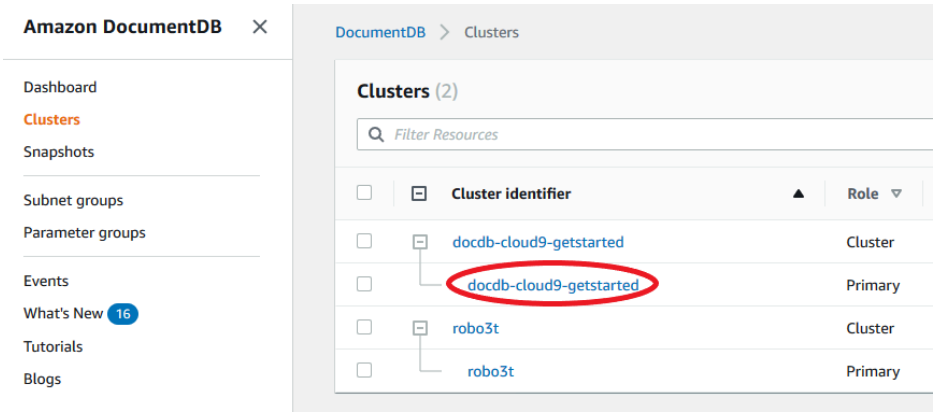
Using the AWS-Managementkonsole

Das folgende Verfahren startet eine Instance neu, die Sie unter Verwendung der Konsole angeben.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Klicken Sie im Navigationsbereich auf Cluster.

> **Tip**
>
> Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Im Cluster-Navigationsfeld sehen Sie die Spalte Cluster-ID. Ihre Instances werden unter Clustern aufgeführt, ähnlich wie im folgenden Screenshot.



4. Markieren Sie das Kästchen links neben der Instance, die Sie neu starten möchten.
5. Wählen Sie Actions (Aktionen), Reboot (Neustart) und dann Reboot (Neustart)aus, um den Neustart zu bestätigen.

Es dauert einige Minuten, bis Ihre Instance neu gestartet wird. Sie können die Instance nur verwenden, wenn ihr Status available ist. Sie können mit der Konsole oder der AWS CLI den Status der Instance überwachen. Weitere Informationen finden Sie unter Überwachung des Status einer Amazon DocumentDB-Instance.

Using the AWS CLI

Um eine Amazon DocumentDB DocumentDB-Instance neu zu starten, verwenden Sie den `reboot-db-instance` Vorgang mit dem `--db-instance-identifier` Parameter. Dieser Parameter gibt den Bezeichner für die Instance an, die neu gestartet werden soll.

Der folgende Code startet die Instance `sample-instance` neu.

Example

Für Linux, macOS oder Unix:

```
aws docdb reboot-db-instance \
       --db-instance-identifier sample-instance
```

Für Windows:

```
aws docdb reboot-db-instance ^
       --db-instance-identifier sample-instance
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "sample-instance",
        "DBInstanceClass": "db.r5.large",
        "Engine": "docdb",
        "DBInstanceStatus": "rebooting",
        "Endpoint": {
            "Address": "sample-instance.node.us-east-1.docdb.amazonaws.com",
            "Port": 27017,
            "HostedZoneId": "ABCDEFGHIJKLM"
        },
        "InstanceCreateTime": "2020-03-27T08:05:56.314Z",
        "PreferredBackupWindow": "02:00-02:30",
        "BackupRetentionPeriod": 1,
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abcd0123",
                "Status": "active"
            }
        ],
        "AvailabilityZone": "us-east-1c",
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-abcd0123",
            "SubnetGroupStatus": "Complete",
            "Subnets": [
                {
                    "SubnetIdentifier": "subnet-abcd0123",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1a"
                    },
```

```
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-wxyz0123",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1b"
                    },
                    "SubnetStatus": "Active"
                }
            ]
        },
        "PreferredMaintenanceWindow": "sun:06:53-sun:07:23",
        "PendingModifiedValues": {},
        "EngineVersion": "3.6.0",
        "AutoMinorVersionUpgrade": true,
        "PubliclyAccessible": false,
        "DBClusterIdentifier": "sample-cluster",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/sample-key",
        "DbiResourceId": "db-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
        "CACertificateIdentifier": "rds-ca-2019",
        "PromotionTier": 1,
        "DBInstanceArn": "arn:aws:rds:us-east-1:<accountID>:db:sample-instance",
        "EnabledCloudwatchLogsExports": [
            "profiler"
        ]
    }
}
```

Es dauert einige Minuten, bis Ihre Instance neu gestartet wird. Sie können die Instance nur verwenden, wenn ihr Status available ist. Sie können mit der Konsole oder der AWS CLI den Status der Instance überwachen. Weitere Informationen finden Sie unter Überwachung des Status einer Amazon DocumentDB-Instance.

Löschen einer Amazon DocumentDB DocumentDB-Instance

Sie können Ihre Amazon DocumentDB DocumentDB-Instance entweder mit dem AWS-Managementkonsole oder dem AWS CLI löschen. Nur Instances im Status available können gelöscht werden. Eine angehaltene Instance kann nicht gelöscht werden. Wenn der Amazon DocumentDB-Cluster, der Ihre Instance enthält, gestoppt ist, starten Sie zuerst den Cluster, warten Sie, bis die

Instance verfügbar ist, und löschen Sie dann die Instance. Weitere Informationen finden Sie unter Stoppen und Starten eines Amazon DocumentDB-Clusters.

> **Note**
>
> Amazon DocumentDB speichert all Ihre Daten im Cluster-Volume. Die Daten bleiben in diesem Cluster-Volume erhalten. Dies gilt auch dann, wenn Sie alle Instances von Ihrem Cluster entfernen. Wenn Sie erneut auf die Daten zugreifen müssen, können Sie jederzeit dem Cluster eine Instance hinzufügen und da fortfahren, wo Sie aufgehört haben.

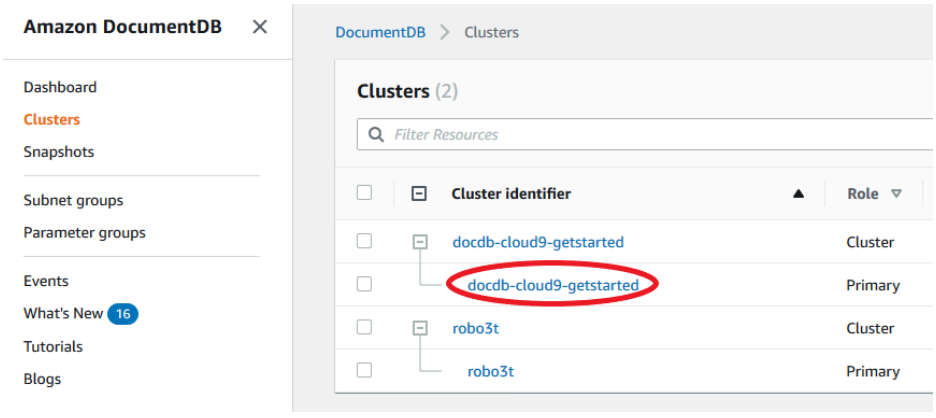
Using the AWS-Managementkonsole

Das folgende Verfahren löscht eine angegebene Amazon DocumentDB DocumentDB-Instance mithilfe der Konsole.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Klicken Sie im Navigationsbereich auf Cluster.

   > **Tip**
   >
   > Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Im Cluster-Navigationsfeld sehen Sie die Spalte Cluster-ID. Ihre Instances werden unter Clustern aufgeführt, ähnlich wie im folgenden Screenshot.



4. Markieren Sie das Kästchen links neben der Instance, die Sie löschen möchten.

5. Wählen Sie Actions (Aktionen) und anschließend Delete (Löschen) aus.

    1. Wenn Sie die letzte Instance in Ihrem Cluster löschen:

        - Create final cluster snapshot? (Endgültigen Cluster-Snapshot erstellen?) — Wählen Sie Ja, wenn Sie einen endgültigen Snapshot erstellen möchten, bevor der Cluster gelöscht wird. Wählen Sie andernfalls No (Nein) aus.
        - Name des endgültigen Snapshots — Wenn Sie einen endgültigen Snapshot erstellen möchten, geben Sie die Cluster-Snapshot-ID des neu erstellten Cluster-Snapshots ein.
        - Delete <instance-name> instance? (Instance <instance-name> löschen?) — Geben Sie den Ausdruck Gesamten Cluster löschen in das Feld ein, um das Löschen zu bestätigen.

    2. Wenn Sie nicht die letzte Instance in Ihrem Cluster löschen:

        - Delete <instance-name> instance? (Instance <instance-name> löschen?) — Geben Sie den Ausdruck delete me in das Feld ein, um das Löschen zu bestätigen.

6. Wählen Sie Delete (Löschen), um die Instance zu löschen.

Es dauert einige Minuten, bis die Instance gelöscht ist. Informationen zum Überwachen des Status einer Instance finden Sie unter Überwachung des Status einer Amazon DocumentDB-Instance.

Using the AWS CLI

Das folgende Verfahren löscht eine Amazon DocumentDB DocumentDB-Instance mithilfe von. AWS CLI

1. Stellen Sie zunächst fest, wie viele Instances sich in Ihrem Amazon DocumentDB-Cluster befinden:

    Führen Sie den `describe-db-clusters`-Befehl wie folgt aus, um festzustellen, wie viele Instances sich in Ihrem Cluster befinden.

    ```
    aws docdb describe-db-clusters \
        --db-cluster-identifier sample-cluster \
        --query 'DBClusters[*].
    [DBClusterIdentifier,DBClusterMembers[*].DBInstanceIdentifier]'
    ```

    Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
[
    [
        "sample-cluster",
        [
            "sample-instance-1",
            "sample-instance-2"
        ]
    ]
]
```

2. Wenn es mehr als eine Instance in Ihrem Amazon DocumentDB-Cluster gibt:

   Um eine angegebene Amazon DocumentDB DocumentDB-Instance zu löschen, verwenden Sie den `delete-db-instance` Befehl mit dem `--db-instance-identifier` Parameter, wie unten gezeigt. Es dauert einige Minuten, bis die Instance gelöscht ist. Informationen zum Überwachen des Status einer Instance finden Sie unter Überwachung des Status einer Amazon DocumentDB-Instance.

   ```
   aws docdb delete-db-instance \
          --db-instance-identifier sample-instance-2
   ```

   Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

   ```
   {
       "DBInstance": {
           "DBInstanceIdentifier": "sample-instance-2",
           "DBInstanceClass": "db.r5.large",
           "Engine": "docdb",
           "DBInstanceStatus": "deleting",
           "Endpoint": {
               "Address": "sample-instance-2.node.us-east-1.docdb.amazonaws.com",
               "Port": 27017,
               "HostedZoneId": "ABCDEFGHIJKLM"
           },
           "InstanceCreateTime": "2020-03-27T08:05:56.314Z",
           "PreferredBackupWindow": "02:00-02:30",
           "BackupRetentionPeriod": 1,
           "VpcSecurityGroups": [
               {
                   "VpcSecurityGroupId": "sg-abcd0123",
                   "Status": "active"
   ```

```
            }
        ],
        "AvailabilityZone": "us-east-1c",
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-6242c31a",
            "SubnetGroupStatus": "Complete",
            "Subnets": [
                {
                    "SubnetIdentifier": "subnet-abcd0123",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1a"
                    },
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-wxyz0123",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1b"
                    },
                    "SubnetStatus": "Active"
                }
            ]
        },
        "PreferredMaintenanceWindow": "sun:06:53-sun:07:23",
        "PendingModifiedValues": {},
        "EngineVersion": "3.6.0",
        "AutoMinorVersionUpgrade": true,
        "PubliclyAccessible": false,
        "DBClusterIdentifier": "sample-cluster",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/sample-key",
        "DbiResourceId": "db-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
        "CACertificateIdentifier": "rds-ca-2019",
        "PromotionTier": 1,
        "DBInstanceArn": "arn:aws:rds:us-east-1:<accountID>:db:sample-instance-2",
        "EnabledCloudwatchLogsExports": [
            "profiler"
        ]
    }
}
```

3. Wenn die Instance, die Sie löschen möchten, die letzte Instance in Ihrem Amazon DocumentDB-Cluster ist:

   Wenn Sie die letzte Instance in einem Amazon DocumentDB-Cluster löschen, löschen Sie auch diesen Cluster und die mit diesem Cluster verknüpften automatischen Snapshots und kontinuierlichen Backups.

   Um die letzte Instance in Ihrem Cluster zu löschen, können Sie den Cluster löschen und optional einen endgültigen Snapshot erstellen. Weitere Informationen finden Sie unter Löschen eines Amazon DocumentDB-Clusters.

Löschschutz

Wenn Sie die letzte Instanz eines Amazon DocumentDB-Clusters löschen, werden auch der Cluster sowie die mit diesem Cluster verknüpften automatischen Snapshots und kontinuierlichen Backups gelöscht. Amazon DocumentDB erzwingt den Löschschutz für einen Cluster, unabhängig davon, ob Sie den Löschvorgang mit AWS-Managementkonsole oder dem ausführen. AWS CLI Sie können einen Cluster nicht löschen, solange der Löschschutz aktiviert ist.

Um einen Cluster mit aktiviertem Löschschutz zu löschen, müssen Sie zuerst den Cluster ändern und den Löschschutz deaktivieren. Weitere Informationen finden Sie unter Löschen eines Amazon DocumentDB-Clusters.

Instanzklassen verwalten

Die Instance-Klasse bestimmt die Rechen- und Speicherkapazität einer Amazon DocumentDB-Instance (mit MongoDB-Kompatibilität). Welche Instance-Klasse Sie benötigen, hängt von der benötigten Rechenleistung und dem Speicherbedarf ab.

Amazon DocumentDB unterstützt die Instance-Klassen R4, R5, R6G, R8G, T3 und T4G. Diese Klassen sind Instance-Klassen der aktuellen Generation, die für speicherintensive Anwendungen optimiert sind. Die Spezifikationen dieser Klassen finden Sie unter Spezifikationen der Instanzklasse.

Themen

- Bestimmung einer Instanzklasse
- Die Klasse einer Instance ändern
- Unterstützte Instance-Klassen nach Region
- Spezifikationen der Instanzklasse

Bestimmung einer Instanzklasse

Um die Klasse einer Instanz zu ermitteln, können Sie die `describe-db-instances` AWS CLI Operation AWS-Managementkonsole oder verwenden.

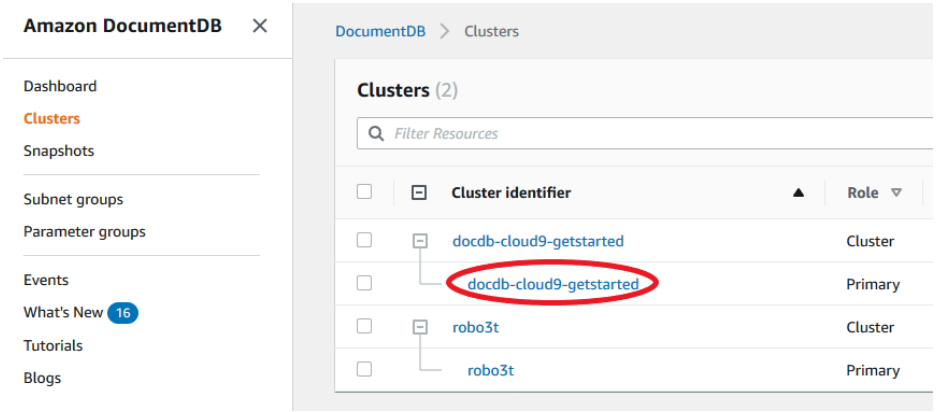
Using the AWS-Managementkonsole

Führen Sie die folgenden Schritte in der Konsole aus, um die Instance-Klasse für die Instances Ihres Clusters zu ermitteln.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Wählen Sie im Navigationsbereich Clusters aus, um die Instance zu finden, an der Sie interessiert sind.

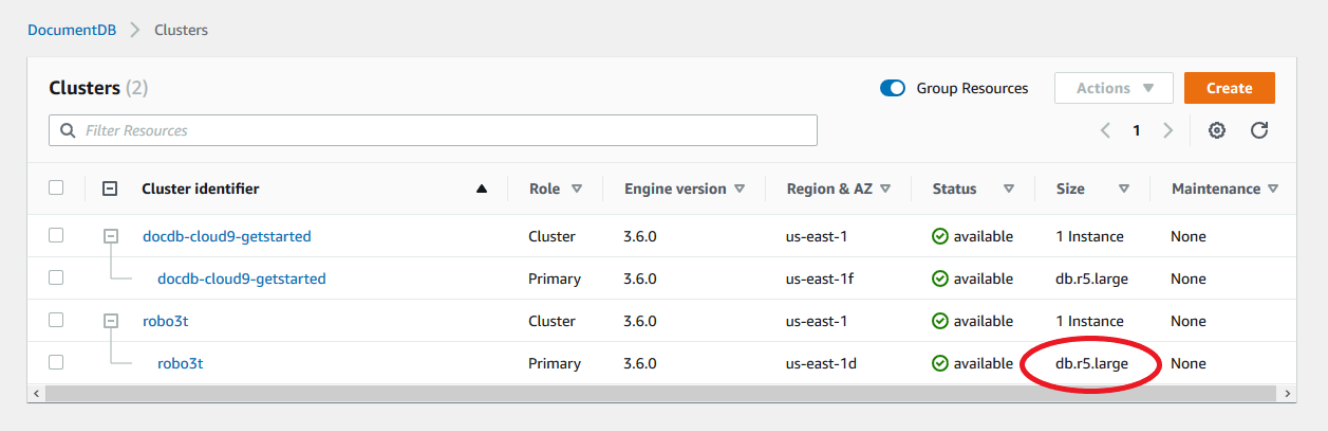
   > **Tip**
   >
   > Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Im Navigationsfeld „Cluster“ wird die Spalte Cluster-ID angezeigt. Ihre Instances sind unter Clustern aufgeführt, ähnlich wie im folgenden Screenshot.



4. Erweitern Sie den Cluster in der Liste der Instances, um die Instances zu finden, an denen Sie interessiert sind. Suchen Sie die gewünschte Instanz. Sehen Sie sich dann die Spalte Größe der Zeile der Instanz an, um die Instanzklasse zu sehen.

   In der folgenden Abbildung ist die Instance-Klasse für die Instance `robo3t` `db.r5.4xlarge`.



Using the AWS CLI

Verwenden Sie die `describe-db-instances` Operation mit den folgenden Parametern AWS CLI, um die Klasse einer Instanz mithilfe von zu ermitteln.

- **`--db-instance-identifier`**— Fakultativ. Gibt die Instance an, für die Sie die Instance-Klasse suchen möchten. Wenn dieser Parameter ausgelassen wird, gibt `describe-db-instances` eine Beschreibung für bis zu 100 Ihrer Instances zurück.
- **`--query`**— Fakultativ. Gibt die Mitglieder der Instance an, die in die Ergebnisse einbezogen werden sollen. Wenn dieser Parameter ausgelassen wird, werden alle Instance-Mitglieder zurückgegeben.

Example

Im folgenden Beispiel werden der Instanzname und die Klasse für die Instanz ermitteltsample-instance-1.

Für Linux, macOS oder Unix:

```
aws docdb describe-db-instances \
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' \
    --db-instance-identifier sample-instance-1
```

Für Windows:

```
aws docdb describe-db-instances ^
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' ^
```

```
    --db-instance-identifier sample-instance-1
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
[
    [
        "sample-instance-1",
        "db.r5.large"
    ]
```

Example

Im folgenden Beispiel werden der Instanzname und die Klasse für bis zu 100 Amazon DocumentDB-Instances ermittelt.

Für Linux, macOS oder Unix:

```
aws docdb describe-db-instances \
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' \
    --filter Name=engine,Values=docdb
```

Für Windows:

```
aws docdb describe-db-instances ^
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' ^
    --filter Name=engine,Values=docdb
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
[
    [
        "sample-instance-1",
        "db.r5.large"
    ],
    [
        "sample-instance-2",
        "db.r5.large"
    ],
    [
        "sample-instance-3",
        "db.r5.4xlarge"
    ],
```

```
    [
        "sample-instance-4",
        "db.r5.4xlarge"
    ]
]
```

Weitere Informationen finden Sie unter Beschreibung von Amazon DocumentDB DocumentDB-Instances.

Die Klasse einer Instance ändern

Sie können die Instanzklasse Ihrer Instanz mit dem AWS-Managementkonsole oder dem ändern AWS CLI. Weitere Informationen finden Sie unter Ändern einer Amazon DocumentDB DocumentDB-Instance.

Unterstützte Instance-Klassen nach Region

Amazon DocumentDB unterstützt die folgenden Instance-Klassen:

- `R8G`— Speicheroptimierte Instances der neuesten Generation mit Arm-based AWS Graviton4-Prozessoren, die im Vergleich zu R6G-Instances eine bis zu 30% bessere Leistung bieten.
- `R6G`— Memory-optimized Instances mit Arm-based AWS Graviton2-Prozessoren, die eine bis zu 30% bessere Leistung als R5-Instances bieten und das bei 5% geringeren Kosten.
- `R6GD`— Memory-optimized R6G-Instances mit lokalem, auf NVMe (NVMe) basierendem Solid-State Drive (SSD) -Speicher für kurzlebige Daten.
- `R5`— Memory-optimized Instances, die bei gleichen Instance-Kosten eine bis zu 100% bessere Leistung als R4-Instances bieten.
- `R4`— Frühere Generation speicheroptimierter Instances.
- `T4G`— Latest-generation kostengünstiger Allzweck-Instance-Typ mit Arm-based AWS Graviton2-Prozessoren, der ein grundlegendes CPU-Leistungsniveau bietet. Er bietet ein bis zu 35% besseres Preis-Leistungs-Verhältnis als T3-Instances und ist ideal für die Ausführung von Anwendungen mit mäßiger CPU-Auslastung, bei denen es zu vorübergehenden Auslastungsspitzen kommt.
- `T3`— Kostengünstiger Allzweck-Instance-Typ, der ein grundlegendes CPU-Leistungsniveau bietet und die Möglichkeit bietet, die CPU-Auslastung jederzeit und so lange wie nötig zu beschleunigen.

Detaillierte Angaben zu den Instance-Klassen finden Sie unter Spezifikationen der Instanzklasse.

> **Note**
>
> I/O-Optimized Speicher ist nur in den Engine-Versionen 5.0 und 8.0 (instanzbasierte Cluster) verfügbar.

Eine bestimmte Instanzklasse kann in einer bestimmten Region unterstützt werden oder auch nicht. Die folgende Tabelle gibt an, welche Instance-Klassen von Amazon DocumentDB in den einzelnen Regionen unterstützt werden.

Unterstützte Instance-Klassen nach Region

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Serverless
USA Ost (Ohio)	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt
USA Ost (Nord-Virginia)	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt
USA West (Oregon)	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt
Afrika (Kapstadt)			Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Südamerika (São Paulo)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Asien-Pazifik (Hongkong)	Unterstützt		Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Asien-Pazifik (Hyderabad)			Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Asien-Pazifik (Malaysia)			Unterstützt			Unterstützt	Unterstützt	Unterstützt

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Serverless
Asien-Pazifik (Mumbai)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Asien-Pazifik (Osaka)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Asien-Pazifik (Seoul)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Asien-Pazifik (Sydney)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Asien-Pazifik (Jakarta)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Asien-Pazifik (Melbourne)			Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Asien-Pazifik (Singapur)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Asien-Pazifik (Thailand)			Unterstützt			Unterstützt	Unterstützt	Unterstützt
Asien-Pazifik (Tokio)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Kanada (Zentral)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Kanada West (Calgary)			Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Europa (Frankfurt)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Serverless
Europa (Zürich)		Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Europa (Irland)	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt	Unterstützt
Europa (London)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Europa (Milan)			Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Europa (Paris)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Europa (Spain)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Europa (Stockholm)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Mexiko (Zentral)			Unterstützt			Unterstützt	Unterstützt	Unterstützt
Naher Osten (VAE)			Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
China (Peking)	Unterstützt	Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
China (Ningxia)	Unterstützt		Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt
Israel (Tel Aviv)			Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Serverless
AWS GovCloud (US-West)	Unterstützt	Unterstützt	Unterstützt	Unterstützt			Unterstützt	Unterstützt
AWS GovCloud (US-East)		Unterstützt	Unterstützt	Unterstützt		Unterstützt	Unterstützt	Unterstützt

Spezifikationen der Instanzklasse

Die folgende Tabelle enthält Einzelheiten zu den Amazon DocumentDB-Instance-Klassen, einschließlich der Instance-Typen, die in jeder Klasse unterstützt werden. Erläuterungen zu den einzelnen Tabellenspalten finden Sie unterhalb der Tabelle.

Instance-Klasse	vCPU[1]	Speicher (GiB) 2	Mehrstufiger NVMe SSD-Cache (GiB) 3	Max. Temperaturspeicher (GiB) 4	Baseline-/Burst-Bandbreite (Gbit/s) 5	Unterstützende Motoren 6
R8G — Memory-Optimized Instance-Klasse der aktuellen Generation, die auf Graviton4 basiert R8G ist die neueste Instance-Familie (Graviton4), die nur in den Engine-Versionen 5.0 und 8.0 verfügbar ist.						
`db.r8g.large`	2	16	-	30	0,937 / 12,5	5.0.0 und 8.0.0
`db.r8g.xlarge`	4	32	-	60	1,875 / 12,5	5.0.0 und 8.0.0
`db.r8g.2xlarge`	8	64	-	121	3,75 / 15,0	5.0.0 und 8.0.0
`db.r8g.4xlarge`	16	128	-	243	7,5 / 15,0	5.0.0 und 8.0.0

Instance-Klasse	vCPU[1]	Speicher (GiB) 2	Mehrstufiger NVMe SSD-Cache (GiB) 3	Max. Temperaturspeicher (GiB) 4	Baseline-/Burst-Bandbreite (Gbit/s) 5	Unterstützende Motoren 6
db.r8g.8xlarge	32	256	-	488	15	5.0.0 und 8.0.0
db.r8g.12xlarge	48	384	-	732	22	5.0.0 und 8.0.0
db.r8g.16xlarge	64	512	-	987	30	5.0.0 und 8.0.0
db.r8g.24xlarge	96	768	-	1484	40	5.0.0 und 8.0.0
db.r8g.48xlarge	192	1536	-	2967	50	5.0.0 und 8.0.0
R6G — Memory-Optimized Instance-Klasse der aktuellen Generation, die auf Graviton2 basiert						
db.r6g.large	2	16	-	32	0,75/10	4.0.0, 5.0.0 und 8.0.0
db.r6g.xlarge	4	32	-	63	1,25/10	4.0.0, 5.0.0 und 8.0.0
db.r6g.2xlarge	8	64	-	126	2,5/10	4.0.0, 5.0.0 und 8.0.0
db.r6g.4xlarge	16	128	-	252	5,0/10	4.0.0, 5.0.0 und 8.0.0
db.r6g.8xlarge	32	256	-	504	12	4.0.0, 5.0.0 und 8.0.0

Instance-Klasse	vCPU[1]	Speicher (GiB) 2	Mehrstufiger NVMe SSD-Cache (GiB) 3	Max. Temperaturspeicher (GiB) 4	Baseline-/Burst-Bandbreite (Gbit/s) 5	Unterstützende Motoren 6
db.r6g.12xlarge	48	384	-	756	20	4.0.0, 5.0.0 und 8.0.0
db.r6g.16xlarge	64	512	-	1008	25	4.0.0, 5.0.0 und 8.0.0
R6GD — Instance-Klasse der aktuellen Generation NVMe-backed , die auf Graviton2 basiert						
db.r6gd.xlarge	4	32	173	64	1,25/10	5.0.0 und 8.0.0
db.r6gd.2xlarge	8	64	346	128	2,5/10	5.0.0 und 8.0.0
db.r6gd.4xlarge	16	128	694	256	5,0/10	5.0.0 und 8.0.0
db.r6gd.8xlarge	32	256	1388	512	12	5.0.0 und 8.0.0
db.r6gd.12xlarge	48	384	2082	768	20	5.0.0 und 8.0.0
db.r6gd.16xlarge	64	512	2776	1024	25	5.0.0 und 8.0.0
R5 — Instance-Klasse der vorherigen Generation Memory-Optimized						
db.r5.large	2	16	-	31	0,75/10	3.6.0, 4.0.0, 5.0.0 und 8.0.0

Instance-Klasse	vCPU[1]	Speicher (GiB) 2	Mehrstufiger NVMe SSD-Cache (GiB) 3	Max. Temperaturspeicher (GiB) 4	Baseline-/Burst-Bandbreite (Gbit/s) 5	Unterstützende Motoren 6
db.r5.xlarge	4	32	-	62	1,25/10	3.6.0, 4.0.0, 5.0.0 und 8.0.0
db.r5.2xlarge	8	64	-	124	2,5/10	3.6.0, 4.0.0, 5.0.0 und 8.0.0
db.r5.4xlarge	16	128	-	249	5,0/10	3.6.0, 4.0.0, 5.0.0 und 8.0.0
db.r5.8xlarge	32	256	-	504	10	3.6.0, 4.0.0, 5.0.0 und 8.0.0
db.r5.12xlarge	48	384	-	748	12	3.6.0, 4.0.0, 5.0.0 und 8.0.0
db.r5.16xlarge	64	512	-	1008	20	3.6.0, 4.0.0, 5.0.0 und 8.0.0
db.r5.24xlarge	96	768	-	1500	25	3.6.0, 4.0.0, 5.0.0 und 8.0.0

R4 — Instance-Klasse der vorherigen Generation Memory-Optimized

R4-Instances werden nur in der Engine-Version 3.6 unterstützt. Beachten Sie, dass Amazon DocumentDB 3.6 am 30. März 2026 das Ende der Standardunterstützung erreicht. Erweiterter Support (kostenpflichtig) ist bis März 2029 verfügbar.

Instance-Klasse	vCPU[1]	Speicher (GiB) 2	Mehrstufiger NVMe SSD-Cache (GiB) 3	Max. Temperaturspeicher (GiB) 4	Baseline-/Burst-Bandbreite (Gbit/s) 5	Unterstützende Motoren 6
db.r4.large	2	15,25	-	30	0,75/10	Nur 3.6.0
db.r4.xlarge	4	30,5	-	60	1,25/10	Nur 3.6.0
db.r4.2xlarge	8	61	-	120	2,5/10	Nur 3.6.0
db.r4.4xlarge	16	122	-	240	5,0 /10	Nur 3.6.0
db.r4.8xlarge	32	244	-	480	10	nur 3.6.0
db.r4.16xlarge	64	488	-	960	25	nur 3.6.0
T4G — Burstable Performance-Instance-Klassen der neuesten Generation, die auf Graviton2 basieren T-series Instances werden im Modus „Unlimited CPU Burst“ ausgeführt. Eine Burst-Nutzung, die über den Basiswert hinausgeht, wird zusätzlich in Rechnung gestellt. Wird für globale Cluster nicht unterstützt.						
db.t4g.medium	2	4	-	8.13	0,256/5	4.0.0, 5.0.0 und 8.0.0
T3 — Instance-Klassen mit Burstable Performance der vorherigen Generation						
db.t3.medium	2	4	-	7,5	0,256/5	3.6.0, 4.0.0, 5.0.0 und 8.0.0

Instance-Klasse	vCPU[1]	Speicher (GiB) 2	Mehrstufiger NVMe SSD-Cache (GiB) 3	Max. Temperaturspeicher (GiB) 4	Baseline-/Burst-Bandbreite (Gbit/s) 5	Unterstützende Motoren 6

1. vCPU — Die Anzahl der virtuellen Zentraleinheiten (CPUs). Eine virtuelle CPU ist eine Kapazitätseinheit, mit der Sie Instance-Klassen vergleichen können. Anstatt einen bestimmten Prozessor für mehrere Monate oder Jahre zu erwerben oder zu leasen, wird jetzt Kapazität stundenweise gemietet. Unser Ziel besteht darin, unabhängig von der tatsächlich zu Grunde liegenden Hardware eine gleichbleibende Menge an CPU-Kapazität zu bieten.
2. Arbeitsspeicher (GiB) — Der RAM in Gigabyte, der der Instanz zugewiesen ist. Häufig ist das Verhältnis zwischen Arbeitsspeicher- und vCPU konsistent.
3. Tiered NVMe SSD Cache — Der in Gigabyte gemessene Speicherplatz auf dem SSD-Volume, der als erweiterter Cache zum Speichern kurzlebiger Daten zugewiesen wird. Dieser Cache ist nur in Instanzen verfügbar. NVMe-backed
4. Max. Temp. storage (GiB) — Der Speicherplatz, gemessen in Gigabyte, der der Instance für den nicht persistenten temporären Dateispeicher zugewiesen ist. Bei NVMe-backed Instances wird dieser Speicher auf einem SSD-Volume gehostet. NVMe-based In allen anderen Fällen wird er im Amazon Elastic Block Store (EBS) gehostet.
5. Baseline-/Burst-Bandbreite (Gbit/s) — Die Burst-Bandbreite stellt die maximale Bandbreite in Gigabit pro Sekunde dar. Teilen Sie durch 8, um den erwarteten Durchsatz in Gigabyte pro Sekunde zu erhalten. Instanzen der Größe 4xlarge und kleiner haben eine Basisbandbreite. Um der zusätzlichen Nachfrage gerecht zu werden, können sie einen I/O Netzwerkgutschriftsmechanismus nutzen, um ihre Basisbandbreite zu überschreiten. Instances können Aufteilungsbandbreite für einen begrenzten Zeitraum verwenden, in der Regel zwischen 5 und 60 Minuten, abhängig von der Instancegröße.
6. Unterstützende Engines — Die Amazon DocumentDB-Engines, die die Instance-Klasse unterstützen.

NVMe-backed Instanzen

NVMe-backed Instances bieten eine bis zu 7-mal schnellere Abfrageleistung für Anwendungen mit großen Datensätzen, die den Speicher einer regulären Instanz überschreiten. Diese Instances nutzen den lokalen NVMe-basierten Solid-State Drive (SSD) -Speicher (Non-Volatile Memory Express), der

auf r6gd-Instances verfügbar ist, um kurzlebige Daten zu speichern, wodurch der netzwerkbasierte Speicherzugriff reduziert und die Leselatenz und der Durchsatz verbessert werden.

Der lokale SSD-Speicherplatz ist in zwei Abschnitte unterteilt:

- Mehrstufiger Cache — Ungefähr 73% der lokalen SSD sind als Datenbank-Cache reserviert, sodass das System bis zu fünfmal mehr Datenbankseiten speichern kann als der Hauptspeicher allein. Die lokale SSD fungiert als Cache der zweiten Ebene, während der vorhandene speicherinterne Puffercache der First-Tier-Cache bleibt. Die Abfrage greift nur auf externen Speicher zu, wenn sowohl der Puffer-Cache als auch der SSD-Cache fehlschlagen.
- Temporärer Speicher — Die verbleibenden 27% sind für den nicht persistenten temporären Dateispeicher reserviert, der für komplexe Abfragen verwendet wird, die Sortierungen oder ressourcenintensive Operationen wie Indexerstellung beinhalten. Bei regulären Instances befindet sich der temporäre Speicherplatz auf einem Amazon Elastic Block Store (EBS) -Volume. Der lokal gehostete temporäre Speicher auf der SSD reduziert die Abfragelatenz durch Sortierungen um das bis zu Zweifache und beschleunigt ressourcenintensive Vorgänge wie Indexerstellungen.

Einzelheiten zum Typ der unterstützten NVMe-backed Instanzen und ihrer Spezifikation finden Sie unter. Spezifikationen der Instanzklasse

Topics

- Empfohlene Anwendungsfälle für NVMe-backed Instanzen
- NVMe-backed Instanzen mit Amazon DocumentDB verwenden
- Instanzen überwachen NVMe-backed

Empfohlene Anwendungsfälle für NVMe-backed Instanzen

Verwenden Sie NVMe-backed Instanzen in den folgenden Szenarien:

- Read-heavy Workloads — Wenn Ihre Arbeitslast leseintensiv ist und Ihr Datensatz größer als der Puffer-Cache ist, was durch niedrige `BufferCacheHitRatio` und hohe `ReadIOPS` Metriken angezeigt wird, können NVMe-backed Instances Leistungsvorteile bieten.
- Update-heavy Workloads — Wenn Ihr Workload updateintensiv ist und die Garbage Collection aufgrund der Leselatenz im Netzwerkspeicher nicht Schritt halten kann, können NVMe-backed Instances helfen, das Problem zu beheben.

NVMe-backed Instanzen können für verschiedene Anwendungsfälle von Vorteil sein, darunter:

- Internet-scale Anwendungen — Anwendungen wie Zahlungsabwicklung, Fakturierung und E-Commerce mit strengen Service Level Agreements (SLAs) zur Leistung können die Leistungsvorteile von NVMe-backed Instances nutzen.
- Real-time Berichts-Dashboards — Dashboards, die Hunderte von Abfragen zur metrics/data Erfassung ausführen, können von der niedrigen Latenz und dem hohen Durchsatz der Instances profitieren. NVMe-backed
- Generative KI-Anwendungen — Anwendungen, die die Vektorsuche verwenden, um in Millionen von Vektoreinbettungen genaue oder nächstgelegene Nachbarn zu finden, können die hohe Leistung von Instances nutzen. NVMe-backed

NVMe-backed Instanzen mit Amazon DocumentDB verwenden

Um NVMe-backed Instanzen von Amazon DocumentDB zu verwenden:

- Erstellen Sie einen Amazon DocumentDB-Cluster und fügen Sie eine der NVMe-backed Instance-Klassen hinzu. Weitere Informationen finden Sie unter Erstellen eines Amazon DocumentDB-Clusters.
- Ändern Sie alternativ einen vorhandenen Amazon DocumentDB-Cluster, um eine der NVMe-backed Instance-Klassen zu verwenden. Weitere Informationen finden Sie unter Ändern eines Amazon DocumentDB-Clusters.

Informationen zur Überprüfung der Verfügbarkeit von NVMe-backed Instances in verschiedenen AWS Regionen finden Sie unterUnterstützte Instance-Klassen nach Region.

Wenn Sie von einer NVMe-backed Instance zu einer regulären Instance zurückkehren möchten, ändern Sie die Datenbank-Instance-Klasse Ihrer Amazon DocumentDB-Instance in eine ähnliche Instance-Klasse ohne NVMe-Speicher. Wenn Ihre aktuelle Instance-Klasse beispielsweise 'db.r6gd.4xlarge' ist, wählen Sie 'db.r6g.4xlarge', um zurück zu wechseln. Weitere Informationen finden Sie unter Ändern eines Amazon DocumentDB-Clusters.

Instanzen überwachen NVMe-backed

Zusätzlich zu den regulären Instance-Metriken, die in Amazon verfügbar sind CloudWatch, geben NVMe-backed Instances zusätzliche Metriken aus, die sich auf NVMe-based SSD-Speicher, IOPS und Durchsatz beziehen.

```
NVMeStorageCacheHitRatio
FreeNVMeStorage
ReadIOPSNVMeStorage
ReadLatencyNVMeStorage
ReadThroughputNVMeStorage
WriteIOPSNVMeStorage
WriteLatencyNVMeStorage
WriteThroughputNVMeStorage
```

Weitere Informationen zu diesen Metriken finden Sie unter NVMe-backed Metriken für Instanzen

Verwaltung von Amazon DocumentDB-Subnetzgruppen

Eine Virtual Private Cloud (VPC) ist ein virtuelles Netzwerk, das nur Ihnen gewidmet ist. AWS-Konto Es ist logisch von anderen virtuellen Netzwerken in der AWS Cloud isoliert. Sie können Ihre AWS Ressourcen, wie z. B. Amazon DocumentDB-Cluster, in Ihrer Amazon VPC starten. Sie können einen IP-Adressbereich für die VPC angeben, Subnetze hinzufügen, Sicherheitsgruppen zuordnen und Routing-Tabellen konfigurieren.

Ein Subnetz ist ein Bereich von IP-Adressen in Ihrer Amazon VPC. Sie können AWS Ressourcen in einem bestimmten Subnetz starten. Verwenden Sie öffentliche Subnetze für Ressourcen, die mit dem Internet verbunden sein müssen. Verwenden Sie private Subnetze für Ressourcen, die nicht mit dem Internet verbunden sein werden. Weitere Informationen zu öffentlichen und privaten Subnetzen finden Sie unter Grundlagen zu VPC und Subnetzen im Amazon Virtual Private Cloud-Benutzerhandbuch.

Eine DB-Subnetzgruppe ist eine Sammlung von Subnetzen, die Sie in einer VPC erstellen, welche Sie dann für Ihre Cluster festlegen. Mithilfe einer Subnetzgruppe können Sie beim Erstellen von Clustern eine bestimmte VPC festlegen. Wenn Sie die `default` Subnetzgruppe verwenden, umfasst diese alle Subnetze in der VPC.

Jede DB-Subnetzgruppe sollte über Subnetze in mindestens zwei Availability Zones in einer bestimmten -Region verfügen. Wenn Sie einen DB-Cluster in einer VPC erstellen, müssen Sie eine DB-Subnetzgruppe auswählen. Amazon DocumentDB verwendet diese DB-Subnetzgruppe und Ihre bevorzugte Availability Zone, um ein Subnetz und eine IP-Adresse innerhalb dieses Subnetzes auszuwählen, die Sie Ihrem Cluster zuordnen möchten. Wenn die primäre Instance ausfällt, kann Amazon DocumentDB eine entsprechende Replikat-Instance zur neuen primären Instance heraufstufen. Anschließend kann mithilfe einer IP-Adresse aus dem Subnetz, in dem sich der vorherige primäre Cluster befand, eine neue Replikat-Instance erstellt werden.

Wenn Amazon DocumentDB eine Instance in einer VPC erstellt, weist es Ihrem Cluster mithilfe einer aus Ihrer DB-Subnetzgruppe ausgewählten IP-Adresse eine Netzwerkschnittstelle zu. Verwenden Sie immer den DNS-Namen, da sich die zugrunde liegende IP-Adresse während eines Failovers ändern kann. Weitere Informationen finden Sie unter Amazon DocumentDB-Endpunkte.

Informationen zum Erstellen Ihrer eigenen VPC und Subnetze finden Sie unter Arbeiten mit VPCs und Subnetzen im Amazon Virtual Private Cloud-Benutzerhandbuch.

Themen

- Erstellen einer Amazon DocumentDB-Subnetzgruppe
- Beschreibung einer Amazon DocumentDB-Subnetzgruppe
- Ändern einer Amazon DocumentDB-Subnetzgruppe
- Löschen einer Amazon DocumentDB-Subnetzgruppe

Erstellen einer Amazon DocumentDB-Subnetzgruppe

Wenn Sie einen Amazon DocumentDB-Cluster erstellen, müssen Sie eine Amazon VPC und die entsprechende Subnetzgruppe innerhalb dieser Amazon VPC auswählen, um Ihren Cluster zu starten. Subnetze bestimmen die Verfügbarkeitszone und den IP-Bereich innerhalb der Verfügbarkeitszone, die Sie zum Starten einer Instance verwenden möchten.

Eine Subnetzgruppe ist eine benannte Gruppe von Subnetzen (oder AZs), mit der Sie die Verfügbarkeitszonen angeben können, die Sie für den Start von Amazon DocumentDB-Instances verwenden möchten. In einem Cluster mit drei Instances wird beispielsweise empfohlen, dass jede dieser Instances in separaten AZs bereitgestellt wird. Dadurch wird die Hochverfügbarkeit optimiert. Wenn also eine einzelne AZ ausfällt, wirkt sich dies nur auf eine einzelne Instanz aus.

Amazon DocumentDB-Instances können in bis zu drei Availability Zones bereitgestellt werden. Selbst wenn eine Subnetzgruppe mehr als drei Subnetze hat, können Sie nur drei dieser Subnetze verwenden, um einen Amazon DocumentDB-Cluster zu erstellen. Wenn Sie eine Subnetzgruppe erstellen, wählen Sie nur die drei Subnetze aus, in denen Sie Ihre Instances bereitstellen möchten.

Beispiel: Ein Cluster wird erstellt und Amazon DocumentDB wählt die AZs {1A, 1B und 1C} aus. Wenn Sie versuchen, eine Instance in AZ {1D} zu erstellen, schlägt der API-Aufruf fehl. Wenn Sie sich jedoch dafür entscheiden, eine Instance zu erstellen, ohne die bestimmte AZ anzugeben, wählt Amazon DocumentDB eine AZ in Ihrem Namen aus. Amazon DocumentDB verwendet einen Algorithmus für den Lastenausgleich der Instances zwischen den AZs, sodass Sie eine hohe

Verfügbarkeit erreichen. Wenn drei Instances bereitgestellt werden, werden sie standardmäßig über drei AZs bereitgestellt und nicht alle in einer einzigen AZ.

Bewährte Methoden

- Wenn Sie keinen speziellen Grund haben, legen Sie immer eine Subnetzgruppe mit drei Subnetzen an. Dadurch wird sichergestellt, dass Cluster mit drei oder mehr Instances eine höhere Verfügbarkeit erreichen können, da Instances über drei AZs bereitgestellt werden.
- Verteilen Sie Instances immer über mehrere AZs, um eine hohe Verfügbarkeit zu erreichen. Platzieren Sie niemals alle Instances für einen Cluster in einer einzigen AZ.
- Da Failover-Ereignisse jederzeit auftreten können, sollten Sie nicht davon ausgehen, dass sich eine primäre Instance oder Replikat-Instance immer in einer bestimmten AZ befinden.

Wie erstelle ich eine Subnetzgruppe

Sie können AWS-Managementkonsole oder verwenden, AWS CLI um eine Amazon DocumentDB-Subnetzgruppe zu erstellen:

Using the AWS-Managementkonsole

Gehen Sie wie folgt vor, um eine Amazon DocumentDB-Subnetzgruppe zu erstellen.

So erstellen Sie eine Amazon-DocumentDB-Subnetzgruppe

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Wählen Sie im Navigationsbereich Subnet groups (Subnetzgruppen) und anschließend Create (Erstellen) aus.

   > **Tip**
   >
   > Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Auf der Seite Create subnet group (Subnetzgruppe erstellen):

   a. Im Abschnitt Subnet group details (Subnetzgruppen-Details:

i. Name — Geben Sie einen aussagekräftigen Namen für die Subnetzgruppe ein.

ii. Beschreibung – Geben Sie eine Beschreibung für die Subnetzgruppe ein.

b. Im Abschnitt Add subnets (Subnetze hinzufügen):

i. VPC — Wählen Sie in der Liste eine VPC für diese Subnetzgruppe aus.

ii. Führen Sie eine der folgenden Aktionen aus:

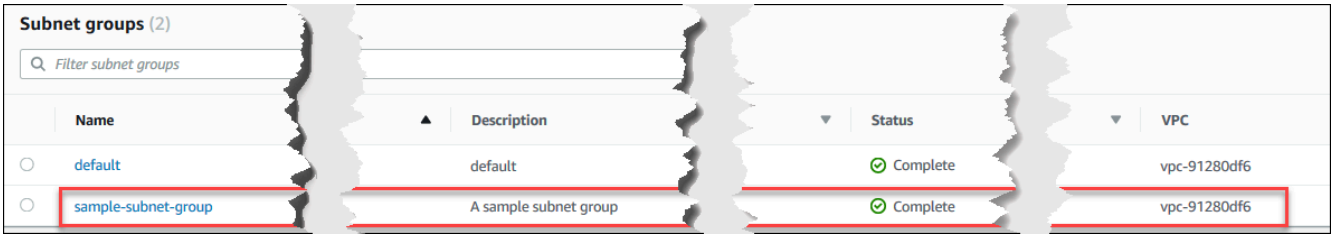
- Um alle Subnetze in der ausgewählten VPC miteinzuschließen, wählen Sie Add all the subnets related to this VPC (Alle zu dieser VPC gehörenden Subnetze hinzufügen).
- Um Subnetze für diese Subnetzgruppe festzulegen, führen Sie folgende Aufgaben für jede Availability Zone aus, für die Sie Subnetze miteinschließen möchten. Sie müssen mindestens zwei Availability Zones einschließen.

A. Verfügbarkeitszone — Wählen Sie in der Liste eine Availability Zone aus.

B. Subnetz — Wählen Sie in der Liste ein Subnetz aus der ausgewählten Availability Zone für diese Subnetzgruppe aus.

C. Wählen Sie Add subnet (Subnetz hinzufügen) aus.

4. Wählen Sie Erstellen aus. Wenn die Subnetzgruppe erstellt wurde, wird sie mit Ihren anderen Subnetzgruppen aufgeführt.



Using the AWS CLI

Bevor Sie mithilfe von eine Subnetzgruppe erstellen können, müssen Sie zunächst ermitteln AWS CLI, welche Subnetze verfügbar sind. Führen Sie den folgenden AWS CLI Vorgang aus, um die Availability Zones und ihre Subnetze aufzulisten.

Parameter:

- **`--db-subnet-group`**— Fakultativ. Bei Angabe einer bestimmten Subnetzgruppe werden die Availability Zones und Subnetze für diese Gruppe aufgelistet. Durch Auslassen dieses

Parameters werden Availability Zones und Subnetze für alle Ihre Subnetzgruppen aufgelistet. Bei Angabe der `default`-Subnetzgruppe werden alle Subnetze der VPC aufgelistet.

Example

Für Linux, macOS oder Unix:

```
aws docdb describe-db-subnet-groups \
    --db-subnet-group-name default \
    --query 'DBSubnetGroups[*].[DBSubnetGroupName,Subnets[*].
[SubnetAvailabilityZone.Name,SubnetIdentifier]]'
```

Für Windows:

```
aws docdb describe-db-subnet-groups ^
    --db-subnet-group-name default ^
    --query 'DBSubnetGroups[*].[DBSubnetGroupName,Subnets[*].
[SubnetAvailabilityZone.Name,SubnetIdentifier]]'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
[
    [
        "default",
        [
            [
                "us-east-1a",
                "subnet-4e26d263"
            ],
            [
                "us-east-1c",
                "subnet-afc329f4"
            ],
            [
                "us-east-1e",
                "subnet-b3806e8f"
            ],
            [
                "us-east-1d",
                "subnet-53ab3636"
            ],
```

```
            [
                "us-east-1b",
                "subnet-991cb8d0"
            ],
            [
                "us-east-1f",
                "subnet-29ab1025"
            ]
        ]
    ]
]
```

Sie können eine neue Subnetzgruppe erstellen, indem Sie die Ausgabe des vorherigen Vorgangs verwenden. Die neue Subnetzgruppe muss über Subnetze aus mindestens zwei Availability Zones verfügen.

Parameter:

- **`--db-subnet-group-name`**—Erforderlich. Der Name für diese Subnetzgruppe.
- **`--db-subnet-group-description`**—Erforderlich. Beschreibung dieser Subnetzgruppe.
- **`--subnet-ids`**—Erforderlich. Eine Liste der Subnetze in dieser Subnetzgruppe. Beispiel: `subnet-53ab3636`.
- --Schlagworte —Optional. Eine Liste der Tags (Schlüssel-Wert-Paare) zum Anfügen an diese Subnetzgruppe.

Der folgende Code erstellt die Subnetzgruppe `sample-subnet-group` mit drei Subnetzen, `subnet-4e26d263`, `subnet-afc329f4` und `subnet-b3806e8f`.

Für Linux, macOS oder Unix:

```
aws docdb create-db-subnet-group \
    --db-subnet-group-name sample-subnet-group \
    --db-subnet-group-description "A sample subnet group" \
    --subnet-ids subnet-4e26d263 subnet-afc329f4 subnet-b3806e8f \
    --tags Key=tag1,Value=One Key=tag2,Value=2
```

Für Windows:

```
aws docdb create-db-subnet-group ^
```

```
    --db-subnet-group-name sample-subnet-group ^
    --db-subnet-group-description "A sample subnet group" ^
    --subnet-ids subnet-4e26d263 subnet-afc329f4 subnet-b3806e8f ^
    --tags Key=tag1,Value=One Key=tag2,Value=2
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    "DBSubnetGroup": {
        "DBSubnetGroupDescription": "A sample subnet group",
        "DBSubnetGroupName": "sample-subnet-group",
        "Subnets": [
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1a"
                },
                "SubnetIdentifier": "subnet-4e26d263",
                "SubnetStatus": "Active"
            },
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1c"
                },
                "SubnetIdentifier": "subnet-afc329f4",
                "SubnetStatus": "Active"
            },
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1e"
                },
                "SubnetIdentifier": "subnet-b3806e8f",
                "SubnetStatus": "Active"
            }
        ],
        "VpcId": "vpc-91280df6",
        "DBSubnetGroupArn": "arn:aws:rds:us-east-1:123SAMPLE012:subgrp:sample-
subnet-group",
        "SubnetGroupStatus": "Complete"
    }
}
```

Beschreibung einer Amazon DocumentDB-Subnetzgruppe

Sie können die AWS-Managementkonsole oder die verwenden AWS CLI , um die Details einer Amazon DocumentDB-Subnetzgruppe abzurufen.

Using the AWS-Managementkonsole

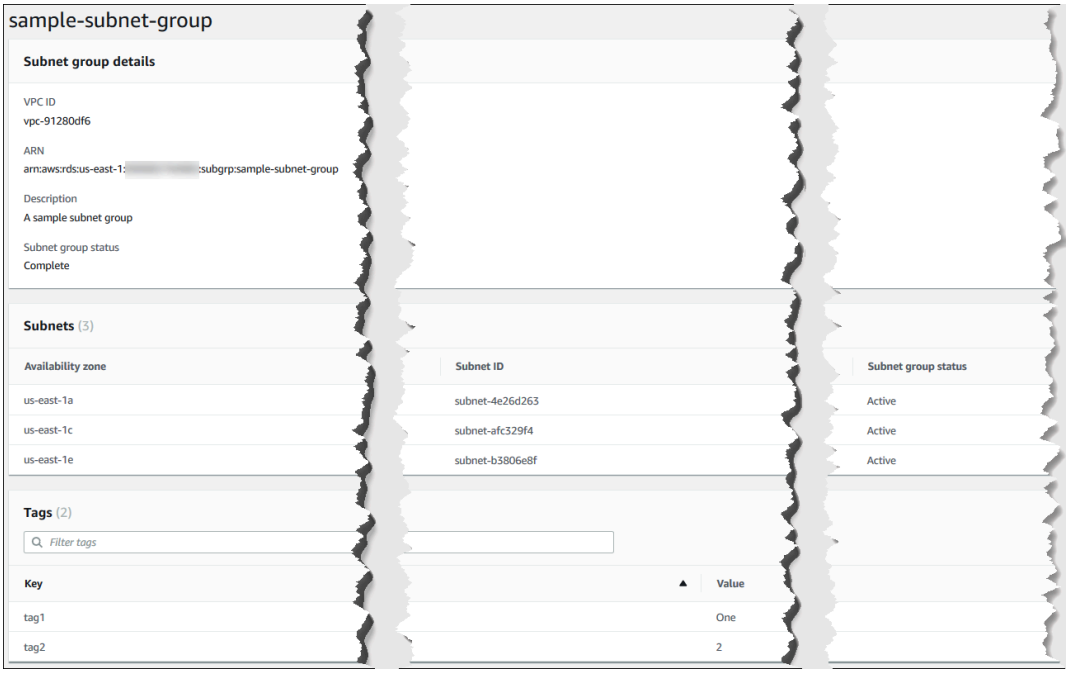
Das folgende Verfahren zeigt Ihnen, wie Sie die Details einer Amazon DocumentDB-Subnetzgruppe abrufen.

So suchen Sie die Details einer Subnetzgruppe

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Wählen Sie im Navigationsbereich Subnetzgruppe aus.

> **Tip**
>
> Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Um die Details einer Subnetzgruppe anzuzeigen, wählen Sie den Namen der Subnetzgruppe.



Using the AWS CLI

Um die Details einer Amazon DocumentDB-Subnetzgruppe zu finden, verwenden Sie den `describe-db-subnet-groups` Vorgang mit dem folgenden Parameter.

Parameter

- `--db-subnet=group-name`— Fakultativ. Falls angegeben, werden Details für die benannte Subnetzgruppe aufgelistet. Falls nicht angegeben, werden Details für bis zu 100 Subnetzgruppen aufgelistet.

Example

Der folgende Code listet die Details für die Subnetzgruppe `sample-subnet-group` auf, die im Abschnitt Erstellen einer Amazon DocumentDB-Subnetzgruppe erstellt wurde.

Für Linux, macOS oder Unix:

```
aws docdb describe-db-subnet-groups \
    --db-subnet-group-name sample-subnet-group
```

Für Windows:

```
aws docdb describe-db-subnet-groups ^
    --db-subnet-group-name sample-subnet-group
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    "DBSubnetGroup": {
        "DBSubnetGroupArn": "arn:aws:rds:us-east-1:123SAMPLE012:subgrp:sample-
subnet-group",
        "VpcId": "vpc-91280df6",
        "SubnetGroupStatus": "Complete",
        "DBSubnetGroupName": "sample-subnet-group",
        "Subnets": [
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1a"
                },
```

```
                "SubnetStatus": "Active",
                "SubnetIdentifier": "subnet-4e26d263"
            },
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1c"
                },
                "SubnetStatus": "Active",
                "SubnetIdentifier": "subnet-afc329f4"
            },
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1e"
                },
                "SubnetStatus": "Active",
                "SubnetIdentifier": "subnet-b3806e8f"
            }
        ],
        "DBSubnetGroupDescription": "A sample subnet group"
    }
}
```

Ändern einer Amazon DocumentDB-Subnetzgruppe

Sie können das AWS-Managementkonsole oder verwenden AWS CLI , um die Beschreibung einer Subnetzgruppe zu ändern oder Subnetze zu einer Amazon DocumentDB-Subnetzgruppe hinzuzufügen oder daraus zu entfernen. Sie können die `default`-Subnetzgruppe jedoch nicht ändern.

Using the AWS-Managementkonsole

Sie können das verwenden AWS-Managementkonsole , um die Beschreibung einer Subnetzgruppe zu ändern oder Subnetze hinzuzufügen oder zu entfernen. Denken Sie daran, dass beim Abschluss des Vorgangs mindestens zwei Availability Zones mit Ihrer Subnetzgruppe verbunden sein müssen.

So ändern Sie Ihre Subnetzgruppe

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb

2. Wählen Sie im Navigationsbereich Subnetzgruppe aus. Klicken Sie dann auf die Schaltfläche links neben dem Namen der Subnetzgruppe. Denken Sie daran, dass Sie die `default`-Subnetzgruppe nicht ändern können.

   > **Tip**
   >
   > Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Wählen Sie Actions (Aktionen) und dann Modify (Ändern) aus.
4. Beschreibung — Um die Beschreibung Ihrer Subnetzgruppe zu ändern, geben Sie eine neue Beschreibung ein.
5. Wenn Sie die mit Ihrer Subnetzgruppe verbundenen Subnetze ändern möchten, führen Sie im Abschnitt Add subnets (Subnetze hinzufügen) eine oder mehrere der folgenden Aktionen durch:
   - Um alle Subnetze aus dieser Subnetzgruppe zu entfernen, wählen Sie Remove all (Alle entfernen) aus.
   - Wenn Sie bestimmte Subnetze aus dieser Subnetzgruppe entfernen möchten, wählen Sie für jedes einzelne Subnetz, das Sie entfernen möchten, Remove (Entfernen) aus.
   - Wenn Sie alle mit dieser VPC verbundenen Subnetze hinzufügen möchten, wählen Sie Add all the subnets related to this VPC (Alle zu dieser VPC gehörenden Subnetze hinzufügen) aus.
   - Wenn Sie bestimmte Subnetze zu dieser Subnetzgruppe hinzufügen möchten, führen Sie die folgenden Aufgaben für jede Availability Zone aus, der Sie ein Subnetz hinzufügen möchten.

     a. Verfügbarkeitszone — Wählen Sie in der Liste eine neue Availability Zone aus.

     b. Subnetz — Wählen Sie in der Liste ein Subnetz aus der ausgewählten Availability Zone für diese Subnetzgruppe aus.

     c. Wählen Sie Add subnet (Subnetz hinzufügen) aus.
6. Im Bestätigungsdialogfeld:
   - Wählen Sie Modify (Ändern), um diese Änderungen an der Subnetzgruppe vorzunehmen.
   - Wenn die Subnetzgruppe unverändert bleiben soll, wählen Sie Cancel (Abbrechen) aus.

Using the AWS CLI

Sie können das verwenden AWS CLI , um die Beschreibung einer Subnetzgruppe zu ändern oder Subnetze hinzuzufügen und zu entfernen. Denken Sie daran, dass beim Abschluss des Vorgangs mindestens zwei Availability Zones mit Ihrer Subnetzgruppe verbunden sein müssen. Es ist nicht möglich, die `default`- Subnetzgruppe zu ändern.

Parameter:

- `--db-subnet-group-name`—Erforderlich. Der Name der Amazon DocumentDB-Subnetzgruppe, die Sie ändern.
- `--subnet-ids`—Erforderlich. Eine Liste aller Subnetze, die nach dieser Änderung Teil der Subnetzgruppe sein sollen.

  > **Important**
  >
  > Alle Subnetze, die sich derzeit in der Subnetzgruppe befinden und die nicht in dieser Liste aufgeführt sind, werden aus der Subnetzgruppe entfernt. Wenn Sie Subnetze, die sich derzeit in der Subnetzgruppe befinden, beibehalten möchten, müssen Sie diese in die Liste aufnehmen.

- `--db-subnet-group-description`— Fakultativ. Beschreibung der Subnetzgruppe.

Example

Der folgende Code ändert die Beschreibung und ersetzt die vorhandenen Subnetze durch die Subnetze `subnet-991cb8d0`, `subnet-53ab3636` und `subnet-29ab1025`.

Für Linux, macOS oder Unix:

```
aws docdb modify-db-subnet-group \
    --db-subnet-group-name sample-subnet-group \
    --subnet-ids subnet-991cb8d0 subnet-53ab3636 subnet-29ab1025 \
    --db-subnet-group-description "Modified subnet group"
```

Für Windows:

```
aws docdb modify-db-subnet-group ^
    --db-subnet-group-name sample-subnet-group ^
```

```
    --subnet-ids subnet-991cb8d0 subnet-53ab3636 subnet-29ab1025 ^
    --db-subnet-group-description "Modified subnet group"
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format). Beachten Sie, dass dies die gleiche Subnetzgruppe ist, die im Abschnitt Erstellen einer Amazon DocumentDB-Subnetzgruppe erstellt wurde. Die Subnetze in der Subnetzgruppe werden jedoch durch die in der Operation`modify-db-subnet-group` angegebenen ersetzt.

```
{
    "DBSubnetGroup": {
        "DBSubnetGroupArn": "arn:aws:rds:us-east-1:123SAMPLE012:subgrp:sample-
subnet-group",
        "DBSubnetGroupDescription": "Modified subnet group",
        "SubnetGroupStatus": "Complete",
        "Subnets": [
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1d"
                },
                "SubnetStatus": "Active",
                "SubnetIdentifier": "subnet-53ab3636"
            },
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1b"
                },
                "SubnetStatus": "Active",
                "SubnetIdentifier": "subnet-991cb8d0"
            },
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1f"
                },
                "SubnetStatus": "Active",
                "SubnetIdentifier": "subnet-29ab1025"
            }
        ],
        "VpcId": "vpc-91280df6",
        "DBSubnetGroupName": "sample-subnet-group"
    }
}
```

Löschen einer Amazon DocumentDB-Subnetzgruppe

Sie können AWS-Managementkonsole oder verwenden, AWS CLI um eine Amazon DocumentDB-Subnetzgruppe zu löschen. Sie können jedoch nicht die `default`-Subnetzgruppe löschen.

Using the AWS-Managementkonsole

Sie können das verwenden AWS-Managementkonsole , um eine Subnetzgruppe zu löschen. Die `default`-Subnetzgruppe kann jedoch nicht gelöscht werden.

So löschen Sie eine Subnetzgruppe

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Wählen Sie im Navigationsbereich Subnetzgruppe aus. Klicken Sie dann auf die Schaltfläche links neben dem Namen der Subnetzgruppe. Denken Sie daran, dass Sie die `default`-Subnetzgruppe nicht löschen können.

   > **Tip**
   >
   > Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Wählen Sie Aktionen und anschließend Löschen aus.
4. Im Bestätigungsdialogfeld:
   - Wählen Sie Delete (Löschen), um die Subnetzgruppe zu löschen.
   - Wählen Sie Cancel (Abbrechen), um die Subnetzgruppe zu erhalten.

Using the AWS CLI

Um eine Amazon DocumentDB-Subnetzgruppe mithilfe von zu löschen AWS CLI, verwenden Sie den `delete-db-subnet-group` Vorgang mit dem folgenden Parameter.

Parameter

- `--db-subnet-group-name`—Erforderlich. Der Name der Amazon DocumentDB-Subnetzgruppe, die gelöscht werden soll. Denken Sie daran, dass Sie die `default`-Subnetzgruppe nicht löschen können.

Example

Der folgende Code löscht `sample-subnet-group`.

Für Linux, macOS oder Unix:

```
aws docdb delete-db-subnet-group \
    --db-subnet-group-name sample-subnet-group
```

Für Windows:

```
aws docdb delete-db-subnet-group ^
    --db-subnet-group-name sample-subnet-group
```

Diese Operation erzeugt keine Ausgabe.

Amazon DocumentDB — Hochverfügbarkeit und Replikation

Sie können in Amazon DocumentDB (mit MongoDB-Kompatibilität) eine hohe Verfügbarkeit und Leseskalierung erreichen, indem Sie Replikat-Instances verwenden. Ein einzelner Amazon DocumentDB-Cluster unterstützt eine einzelne primäre Instance und bis zu 15 Replica-Instances. Diese Instances können über mehrere Availability Zones innerhalb der Cluster-Region verteilt werden. Die primäre Instance akzeptiert Lese- und Schreibverkehr, und Replikat-Instances akzeptieren nur Leseanforderungen.

Das Cluster-Volume besteht aus mehreren Kopien der Daten für den Cluster. Die Daten im Cluster-Volume werden jedoch als ein einziges, logisches Volume für die primäre Instance und die Amazon DocumentDB-Repliken im Cluster dargestellt. Replikat-Instances sind „Eventually Consistent“. Sie geben Abfrageergebnisse mit minimaler Replikatverzögerung zurück – im Normalfall beträgt die Verzögerung weniger als 100 Millisekunden, nachdem die primäre Instance eine Aktualisierung geschrieben hat. Die Replica-Verzögerung variiert in Abhängigkeit vom Veränderungsgrad in der Datenbank. Das heißt, in Zeiten, in denen eine große Anzahl von Schreiboperationen für die Datenbank durchgeführt wird, kann es zu einer Erhöhung der Replikationsverzögerung kommen.

Skalierung von Lesevorgängen

Amazon DocumentDB-Replikate eignen sich gut für die Skalierung von Lesevorgängen, da sie vollständig für Lesevorgänge auf Ihrem Cluster-Volume vorgesehen sind. Schreibvorgänge werden von der primären Instance verwaltet. Das Cluster-Volume wird von allen Instances in Ihrem Cluster

gemeinsam genutzt. Daher müssen Sie nicht für jedes Amazon DocumentDB-Replikat eine Kopie der Daten replizieren und verwalten.

Hohe Verfügbarkeit

Wenn Sie einen Amazon DocumentDB-Cluster erstellen, stellt Amazon DocumentDB je nach Anzahl der Availability Zones in der Subnetzgruppe (es müssen mindestens zwei sein) Instances über die Availability Zones hinweg bereit. Wenn Sie Instances im Cluster erstellen, verteilt Amazon DocumentDB die Instances automatisch auf die Availability Zones in einer Subnetzgruppe, um den Cluster auszugleichen. Diese Vorgehensweise verhindert auch, dass sich alle Instances in derselben Availability Zone befinden.

Beispiel

Zur Veranschaulichung nehmen wir ein Beispiel, in dem ein Cluster erstellt wird, das eine Subnetzgruppe mit drei Availability Zones hat AZ1, AZ2 und AZ3.

Wenn die erste Instance im Cluster erstellt wird, ist diese die primäre Instance und befindet sich in einer der Availability Zones. In diesem Beispiel handelt es sich um AZ1. Die zweite erstellte Instance ist eine Replikat-Instance und befindet sich in einer der beiden anderen Availability Zones, z.B. AZ2. Die dritte Instance, die erstellt wird, ist eine Replikat-Instance und befindet sich in der verbleibenden Availability ZoneAZ3. Wenn Sie mehrere Instances erstellen, werden diese über die Availability Zones verteilt, sodass der Cluster ausbalanciert ist.

Tritt in der primären Instance (AZ1) ein Fehler auf, wird ein Failover ausgelöst und eine der bestehenden Replikat-Instances wird zur primären Instance heraufgestuft. Wenn die alten primäre Instance wiederhergestellt wurde, wird sie zu einem Replikat in der gleichen Availability Zone, in der sie bereitgestellt wurde (AZ1). Wenn Sie einen Cluster mit drei Instanzen bereitstellen, behält Amazon DocumentDB diesen Cluster mit drei Instanzen weiterhin bei. Amazon DocumentDB kümmert sich automatisch um die Erkennung, den Failover und die Wiederherstellung von Instance-Ausfällen, ohne dass ein manuelles Eingreifen erforderlich ist.

Wenn Amazon DocumentDB einen Failover durchführt und eine Instance wiederherstellt, verbleibt die wiederhergestellte Instance in der Availability Zone, in der sie ursprünglich bereitgestellt wurde. Die Rolle der Instance ändert sich möglicherweise von der primären Instance zum Replikat. Dadurch wird ein Szenario verhindert, in dem eine Reihe von Failovers in allen Instances auftreten könnten, die sich in derselben Availability Zone befinden.

Sie können Amazon DocumentDB-Repliken als Failover-Ziele angeben. Das heißt, wenn die primäre Instance ausfällt, wird das angegebene Amazon DocumentDB-Replikat oder das Replikat einer

Stufe zur primären Instance hochgestuft. Es gibt dann eine kurze Unterbrechung, während der Lese- und Schreibanfragen an die primäre Instance mit einer Ausnahme fehlschlagen. Wenn Ihr Amazon DocumentDB-Cluster keine Amazon DocumentDB-Replikate enthält, wird die primäre Instance neu erstellt, wenn sie ausfällt. Das Heraufstufen einer Amazon DocumentDB-Replica ist viel schneller als die Neuerstellung der primären Instance.

Für eine hohe Verfügbarkeit erstellen Sie eine oder mehrere Amazon DocumentDB-Repliken derselben Instance-Klasse wie die primäre Instance in verschiedenen Availability Zones.

Weitere Informationen finden Sie hier:

- Grundlegendes zur Amazon DocumentDB-Cluster-Fehlertoleranz
- Amazon DocumentDB-Failover
  - Steuerung des Failover-Ziels

In den Amazon DocumentDB-Versionen 5.0 und 8.0 werden Replikat-Instances nicht neu gestartet, wenn die primäre Instance neu gestartet wird. Sie bearbeiten weiterhin Leseanforderungen bei Neustarts der primären Instance und sorgen so für eine verbesserte Verfügbarkeit.

Hohe Verfügbarkeit mit globalen Clustern

Für eine hohe Verfügbarkeit über mehrere AWS-Regionen hinweg können Sie globale Amazon DocumentDB-Cluster einrichten. Jeder globale Cluster erstreckt sich über mehrere Regionen und ermöglicht so globale Lesevorgänge mit niedriger Latenz und die Notfallwiederherstellung nach Ausfällen in verschiedenen Regionen. Amazon DocumentDB übernimmt automatisch die Replikation aller Daten und Aktualisierungen von der primären Region in jede der sekundären Regionen.

Hinzufügen von -Replicas

Die erste dem Cluster hinzugefügte Instance ist die primäre Instance. Jede Instance, die nach der ersten Instance hinzugefügt wird, ist eine Replikat-Instance. Ein Cluster kann zusätzlich zur primären Instanz über bis zu 15 Replikat-Instances verfügen.

Wenn Sie einen Cluster mithilfe von erstellen AWS-Managementkonsole, wird gleichzeitig automatisch eine primäre Instanz erstellt. Um ein Replikat gleichzeitig mit der Erstellung des Clusters und der primären Instance zu erstellen, wählen Sie Create replica in different zone (Replikat in unterschiedlicher Zone erstellen) aus. Weitere Informationen finden Sie in Schritt 4.d unter Erstellen eines Amazon DocumentDB-Clusters. Informationen zum Hinzufügen weiterer Replikate

zu einem Amazon DocumentDB-Cluster finden Sie unter. Hinzufügen einer Amazon DocumentDB DocumentDB-Instance zu einem Cluster

Wenn Sie den verwenden AWS CLI , um Ihren Cluster zu erstellen, müssen Sie Ihre Primär- und Replikat-Instances explizit erstellen. Weitere Informationen finden Sie im Abschnitt „Verwenden des AWS CLI“ in den folgenden Themen:

- Erstellen eines Amazon DocumentDB-Clusters
- Hinzufügen einer Amazon DocumentDB DocumentDB-Instance zu einem Cluster

Amazon DocumentDB-Failover

In bestimmten Fällen, wie z. B. bei bestimmten geplanten Wartungsarbeiten oder im unwahrscheinlichen Fall eines Ausfalls eines primären Knotens oder einer Availability Zone, erkennt Amazon DocumentDB (mit MongoDB-Kompatibilität) den Ausfall und ersetzt den primären Knoten. Während eines Failovers wird die Ausfallzeit für Schreibvorgänge minimiert. Das liegt daran, dass die Rolle des primären Knotens auf eine der Read Replicas übergeht, statt dass ein neuer primärer Knoten erstellt und bereitgestellt werden muss. Durch Ausfallerkennung und Replikatheraufstufung wird sichergestellt, dass Sie weiter in den neuen primären Knoten schreiben können, sobald die Heraufstufung abgeschlossen wurde.

Damit ein Failover funktioniert, muss Ihr Cluster über mindestens zwei Instances verfügen — eine primäre und mindestens eine Replikat-Instance.

> **Note**
>
> Dieses Thema bezieht sich nur auf originale, auf Amazon DocumentDB-Instances basierende Cluster. Es gilt nicht für elastische oder globale Cluster.

Steuerung des Failover-Ziels

Amazon DocumentDB bietet Ihnen Failover-Tiers, mit denen Sie kontrollieren können, welche Replikat-Instance bei einem Failover zur primären Instanz hochgestuft wird.

Failover-Tiers

Jede Replikatinstanz ist einer Failoverstufe (0—15) zugeordnet. Wenn aufgrund einer Wartung oder eines unwahrscheinlichen Hardwarefehlers ein Failover auftritt, wechselt die primäre Instanz

zu einem Replikat mit der höchsten Priorität (der niedrigsten nummerierten Stufe). Wenn mehrere Replikate dieselbe Prioritätsstufe haben, erfolgt für das primäre Replikat ein Failover auf das Replikat dieser Stufe, das in seiner Größe dem vorherigen primären Replikat am nächsten kommt.

Indem Sie die Failover-Stufe für eine Gruppe ausgewählter Replikate auf `0` (höchste Priorität) setzen, können Sie sicherstellen, dass ein Failover auf eines der Replikate in dieser Gruppe wechselt. Sie können effektiv verhindern, dass bestimmte Replikate im Falle eines Failover zur primären Instance hochgestuft werden, indem Sie diesen Replikaten eine niedrige Stufe (hohe Anzahl) zuweisen. Dies ist nützlich, wenn bestimmte Replikate von einer Anwendung stark genutzt werden und ein Failover auf eine von ihnen eine kritische Anwendung negativ beeinflussen würde.

Wir empfehlen, dass Replikate, die potenzielle Failover-Ziele sind, dieselbe Instance-Klasse wie die primäre Instance verwenden. Dadurch wird eine Beeinträchtigung der Schreibleistung vermieden, wenn ein Failover ein Replikat während Betriebssystemupdates oder anderen Wartungsarbeiten hochstuft.

Sie können die Failover-Stufe einer Instance beim Erstellen oder später festlegen. Das Festlegen einer Instance-Failover-Stufe durch Ändern der Instance löst keinen Failover aus. Weitere Informationen finden Sie in den folgenden Themen:

- Hinzufügen einer Amazon DocumentDB DocumentDB-Instance zu einem Cluster
- Ändern einer Amazon DocumentDB DocumentDB-Instance

Wenn Sie einen Failover manuell einleiten, haben Sie zwei Möglichkeiten zur Steuerung, welche Replikat-Instance auf primär umgestellt wird: die Failover-Stufen wie zuvor beschrieben und den Parameter `--target-db-instance-identifier`.

`– Ziel-DB-Instanz-ID`

Zum Testen können Sie mit der Operation `failover-db-cluster` ein Failover-Ereignis erzwingen. Mit dem Parameter `--target-db-instance-identifier` können Sie festlegen, welches Replikat auf primär umgestellt werden soll. Die Verwendung des Parameters `--target-db-instance-identifier` ersetzt die Failover-Prioritätsstufe. Wenn Sie den Parameter `--target-db-instance-identifier` nicht angeben, entspricht die Primär-Failover-Funktion der Failover-Prioritätsstufe.

Was passiert bei einem Failover

Das Failover wird automatisch von Amazon DocumentDB abgewickelt, sodass Ihre Anwendungen den Datenbankbetrieb so schnell wie möglich ohne administratives Eingreifen wieder aufnehmen können.

- Wenn Sie beim Failover eine Amazon DocumentDB-Replica-Instance in derselben oder einer anderen Availability Zone haben: Amazon DocumentDB dreht den Canonical Name Record (CNAME) für Ihre Instance um, sodass er auf das fehlerfreie Replikat verweist, das wiederum zum neuen primären Replikat befördert wird. Das Failover wird in der Regel innerhalb von 30 Sekunden vom Anfang bis zum Ende abgeschlossen.
- Wenn Sie keine Amazon DocumentDB-Replikatinstanz haben (z. B. einen einzelnen Instance-Cluster): Amazon DocumentDB versucht, eine neue Instance in derselben Availability Zone wie die ursprüngliche Instance zu erstellen. Dieser Austausch der ursprünglichen Instance wird nach bestem Bemühen durchgeführt, ist aber nicht immer erfolgreich, z. B. wenn ein Problem vorliegt, das sich allgemein auf die Availability Zone auswirkt.

Bei Verbindungsunterbrechung muss Ihre Anwendung versuchen, die Verbindung zur Datenbank wiederherzustellen.

Testen eines Failovers

Bei einem Failover für einen Cluster wird eine der Amazon DocumentDB-Repliken (schreibgeschützte Instances) im Cluster zur primären Instance (Cluster-Writer) hochgestuft.

Wenn die primäre Instance ausfällt, führt Amazon DocumentDB automatisch ein Failover auf ein Amazon DocumentDB-Replikat durch, falls eines vorhanden ist. Sie können ein Failover erzwingen, wenn Sie einen Ausfall einer primären Instance zum Testen simulieren möchten. Jede Instance in einem Cluster hat eine eigene Endpunkt-Adresse. Aus diesem Grund müssen Sie alle bestehenden Verbindungen, die diese Endpunktadressen verwenden, bereinigen und wiederherstellen, wenn der Failover abgeschlossen ist.

Um einen Failover zu erzwingen, verwenden Sie die Operation `failover-db-cluster` mit diesen Parametern.

- `--db-cluster-identifier`—Erforderlich. Der Name des Clusters, der einen Failover durchführen soll.

- `--target-db-instance-identifier`— Fakultativ. Der Name der Instance, die zur primären Instance befördert werden soll.

Example

Die folgende Operation erzwingt einen Failover des Clusters `sample-cluster`. Es gibt nicht an, welche Instance die neue primäre Instance sein soll, weshalb Amazon DocumentDB die Instance entsprechend der Priorität der Failover-Tier auswählt.

Für Linux, macOS oder Unix:

```
aws docdb failover-db-cluster \
   --db-cluster-identifier sample-cluster
```

Für Windows:

```
aws docdb failover-db-cluster ^
   --db-cluster-identifier sample-cluster
```

Die folgende Operation erzwingt einen Failover des Clusters `sample-cluster` und legt fest, dass `sample-cluster-instance` in die primäre Rolle befördert werden soll. (Beachten Sie `"IsClusterWriter": true` in der Ausgabe.)

Für Linux, macOS oder Unix:

```
aws docdb failover-db-cluster \
   --db-cluster-identifier sample-cluster \
   --target-db-instance-identifier sample-cluster-instance
```

Für Windows:

```
aws docdb failover-db-cluster ^
   --db-cluster-identifier sample-cluster ^
   --target-db-instance-identifier sample-cluster-instance
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    "DBCluster": {
        "HostedZoneId": "Z2SUY0A1719RZT",
```

```
        "Port": 27017,
        "EngineVersion": "3.6.0",
        "PreferredMaintenanceWindow": "thu:04:05-thu:04:35",
        "BackupRetentionPeriod": 1,
        "ClusterCreateTime": "2018-06-28T18:53:29.455Z",
        "AssociatedRoles": [],
        "DBSubnetGroup": "default",
        "MasterUsername": "master-user",
        "Engine": "docdb",
        "ReadReplicaIdentifiers": [],
        "EarliestRestorableTime": "2018-08-21T00:04:10.546Z",
        "DBClusterIdentifier": "sample-cluster",
        "ReaderEndpoint": "sample-cluster.node.us-east-1.docdb.amazonaws.com",
        "DBClusterMembers": [
            {
                "DBInstanceIdentifier": "sample-cluster-instance",
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1,
                "IsClusterWriter": true
            },
            {
                "DBInstanceIdentifier": "sample-cluster-instance-00",
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1,
                "IsClusterWriter": false
            },
            {
                "DBInstanceIdentifier": "sample-cluster-instance-01",
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1,
                "IsClusterWriter": false
            }
        ],
        "AvailabilityZones": [
            "us-east-1b",
            "us-east-1c",
            "us-east-1a"
        ],
        "DBClusterParameterGroup": "default.docdb3.6",
        "Endpoint": "sample-cluster.node.us-east-1.docdb.amazonaws.com",
        "IAMDatabaseAuthenticationEnabled": false,
        "AllocatedStorage": 1,
        "LatestRestorableTime": "2018-08-22T21:57:33.904Z",
        "PreferredBackupWindow": "00:00-00:30",
```

```
        "StorageEncrypted": false,
        "MultiAZ": true,
        "Status": "available",
        "DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster",
        "VpcSecurityGroups": [
            {
                "Status": "active",
                "VpcSecurityGroupId": "sg-12345678"
            }
        ],
        "DbClusterResourceId": "cluster-ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    }
}
```

Replikationsverzögerung

Die Replikationsverzögerung beträgt in der Regel 50 ms oder weniger. Die häufigsten Gründe für eine erhöhte Replikatverzögerung sind:

- Eine hohe Schreibrate auf der Primärseite, die dazu führt, dass die Lesereplikate hinter die Primärreplikate zurückfallen.
- Konflikt bei den Lesereplikaten zwischen lang andauernden Abfragen (z. B. große sequentielle Scans, Aggregationsabfragen) und eingehenden Schreibreplikationen.
- Sehr große Anzahl gleichzeitiger Abfragen auf den Lesereplikaten.

Probieren Sie die folgenden Techniken zur Problembehandlung aus, um die Verzögerung bei der Replikation zu minimieren:

- Wenn Sie eine hohe Schreibrate oder eine hohe CPU-Auslastung haben, skalieren Sie die Instances in Ihrem Cluster.
- Wenn lange Abfragen auf Ihren Lesereplikaten laufen und die abgefragten Dokumente sehr häufig aktualisiert werden, sollten Sie erwägen, Ihre Abfragen mit langer Laufzeit zu ändern oder sie für das Replikat auszuführen, um Konflikte bei den primary/write Read Replicas zu vermeiden.
- Wenn es eine sehr große Anzahl gleichzeitiger Abfragen oder eine hohe CPU-Auslastung nur für die Read Replicas gibt, besteht eine andere Möglichkeit darin, die Anzahl der Read Replicas zu skalieren, um die Arbeitslast zu verteilen.
- Da Replikationsverzögerungen auf einen hohen Schreibdurchsatz und lang andauernde Abfragen zurückzuführen sind, sollten Sie die Replikationsverzögerung mithilfe der

`DBClusterReplicaLagMaximum` CloudWatch Metrik in Kombination mit dem langsamen Abfrage-Logger und den /-Metriken beheben. `WriteThroughput` `WriteIOPS`

Stellen Sie sicher, dass alle Ihre Replikate vom gleichen Instance-Typ sind, damit ein Cluster-Failover nicht zu Leistungseinbußen führt.

Wenn Sie zwischen Hochskalierung und Out-Skalierung wählen (z. B. sechs kleinere Instances gegenüber drei größeren Instances), empfehlen wir generell, vor dem Scale-Out zuerst eine Hochskalierung durchzuführen (größere Instances), da Sie so einen größeren Puffer-Cache pro DB-Instance erhalten.

Proaktiv sollten Sie einen Alarm für Replikationsverzögerungen einrichten und dessen Schwellenwert auf einen Wert setzen, der Ihrer Meinung nach die Obergrenze dafür ist, wie weit Ihre Daten auf Replikat-Instances zurückliegen (oder „veraltet“) sein können, bevor die Funktionalität Ihrer Anwendung beeinträchtigt wird. Im Allgemeinen empfehlen wir, den Schwellenwert für die Replikationsverzögerung aufgrund vorübergehender Arbeitslasten für mehrere Datenpunkte zu überschreiten, bevor ein Alarm ausgelöst wird.

> **Note**
>
> Stellen Sie einen weiteren Alarm für Replikationsverzögerungen ein, die 10 Sekunden überschreiten. Wenn Sie diesen Schwellenwert für mehrere Datenpunkte überschreiten, skalieren Sie Ihre Instances oder reduzieren Sie Ihren Schreibdurchsatz auf der primären Instance.

Amazon DocumentDB DocumentDB-Ereignisse verwalten

Amazon DocumentDB (mit MongoDB-Kompatibilität) zeichnet Ereignisse auf, die sich auf Ihre Cluster, Instances, Snapshots, Sicherheitsgruppen und Cluster-Parametergruppen beziehen. Diese Informationen beinhalten Datum und Zeit eines Ereignisses, den Quellnamen und Quelltyp des Ereignisses und eine dem Ereignis zugehörige Benachrichtigung.

> **Important**
>
> Für bestimmte Verwaltungsfunktionen verwendet Amazon DocumentDB Betriebstechnologie, die mit Amazon RDS und Amazon Neptune gemeinsam genutzt wird. Regionsgrenzen, Grenzwerte, die auf Regionsebene geregelt werden, werden von Amazon DocumentDB,

Amazon RDS und Amazon Neptune gemeinsam genutzt. Weitere Informationen finden Sie unter Regionale Kontingente.

Themen

- Amazon DocumentDB DocumentDB-Ereigniskategorien anzeigen
- Amazon DocumentDB DocumentDB-Ereignisse anzeigen

Amazon DocumentDB DocumentDB-Ereigniskategorien anzeigen

Jeder Amazon DocumentDB DocumentDB-Ressourcentyp hat bestimmte Ereignistypen, die ihm zugeordnet werden können. Sie können den AWS CLI `describe-event-categories` Vorgang verwenden, um die Zuordnung zwischen Ereignistypen und Amazon DocumentDB DocumentDB-Ressourcentypen anzuzeigen.

Parameters

- **`--source-type`**— Fakultativ. Verwenden Sie den `--source-type`-Parameter, um die Ereigniskategorien für einen bestimmten Quelltyp anzuzeigen. Die folgenden Werte sind zulässig:
  - `db-cluster`
  - `db-instance`
  - `db-parameter-group`
  - `db-security-group`
  - `db-cluster-snapshot`
- **`--filters`**—Fakultativ. Verwenden Sie den Filter`--filter Name=engine,Values=docdb`, um die Ereigniskategorien nur für Amazon DocumentDB anzuzeigen.

Example

Der folgende Code listet die Ereigniskategorien im Zusammenhang mit Clustern auf.

Für Linux, macOS oder Unix:

```
aws docdb describe-event-categories \
    --filter Name=engine,Values=docdb \
    --source-type db-cluster
```

Für Windows:

```
aws docdb describe-event-categories ^
    --filter Name=engine,Values=docdb ^
    --source-type db-cluster
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    "EventCategoriesMapList": [
        {
            "EventCategories": [
                "notification",
                "failure",
                "maintenance",
                "failover"
            ],
            "SourceType": "db-cluster"
        }
    ]
}
```

Der folgende Code listet die Ereigniskategorien auf, die jedem Amazon DocumentDB DocumentDB-Quelltyp zugeordnet sind.

```
aws docdb describe-event-categories
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    "EventCategoriesMapList": [
        {
            "SourceType": "db-instance",
            "EventCategories": [
                "notification",
                "failure",
                "creation",
                "maintenance",
                "deletion",
                "recovery",
                "restoration",
                "configuration change",
```

```
                "read replica",
                "backtrack",
                "low storage",
                "backup",
                "availability",
                "failover"
            ]
        },
        {
            "SourceType": "db-security-group",
            "EventCategories": [
                "configuration change",
                "failure"
            ]
        },
        {
            "SourceType": "db-parameter-group",
            "EventCategories": [
                "configuration change"
            ]
        },
        {
            "SourceType": "db-cluster",
            "EventCategories": [
                "notification",
                "failure",
                "maintenance",
                "failover"
            ]
        },
        {
            "SourceType": "db-cluster-snapshot",
            "EventCategories": [
                "backup"
            ]
        }
    ]
}
```

Amazon DocumentDB DocumentDB-Ereignisse anzeigen

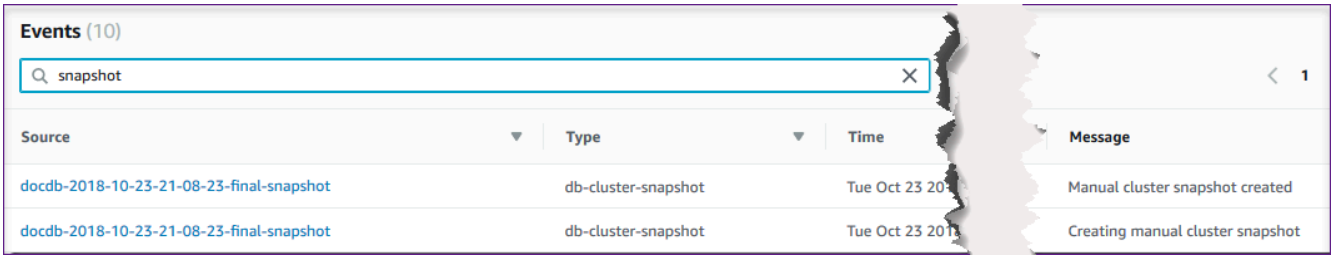
Sie können Ereignisse für Ihre Amazon DocumentDB DocumentDB-Ressourcen über die Amazon DocumentDB DocumentDB-Konsole abrufen, in der Ereignisse der letzten 24 Stunden angezeigt

werden. Sie können auch Ereignisse für Ihre Amazon DocumentDB DocumentDB-Ressourcen abrufen, indem Sie den AWS CLI Befehl describe-events oder den DescribeEventsAmazon DocumentDB DocumentDB-API-Vorgang verwenden. Wenn Sie die AWS CLI oder die Amazon DocumentDB DocumentDB-API verwenden, um Ereignisse anzuzeigen, können Sie Ereignisse der letzten 14 Tage abrufen.

Using the AWS-Managementkonsole

Um alle Amazon DocumentDB DocumentDB-Instance-Ereignisse der letzten 24 Stunden anzuzeigen

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Wählen Sie im Navigationsbereich die Option Events. Die verfügbaren Ereignisse erscheinen in einer Liste.
3. Verwenden Sie die Liste Filter, um die Ereignisse nach Typ zu filtern. Geben Sie einen Begriff in das Textfeld ein, um Ihre Ergebnisse weiter zu filtern. Der folgende Screenshot zeigt beispielsweise, wie alle Amazon DocumentDB DocumentDB-Ereignisse nach Snapshot-Ereignissen gefiltert werden.



Using the AWS CLI

Um alle Amazon DocumentDB DocumentDB-Instance-Ereignisse der letzten 7 Tage anzuzeigen

Sie können alle Amazon DocumentDB DocumentDB-Instance-Ereignisse der letzten 7 Tage anzeigen, indem Sie den AWS CLI Vorgang describe-events ausführen, wobei der `--duration` Parameter auf `10080` (10.080 Minuten) gesetzt ist.

```
aws docdb describe-events --duration 10080
```

Filtern nach Amazon DocumentDB DocumentDB-Ereignissen

Um bestimmte Amazon DocumentDB DocumentDB-Ereignisse zu sehen, verwenden Sie den `describe-events` Vorgang mit den folgenden Parametern.

Parameters

- **`--filter`**— Erforderlich, um zurückgegebene Werte auf Amazon DocumentDB DocumentDB-Ereignisse zu beschränken. Wird verwendet**`Name=engine,Values=docdb`**, um alle Ereignisse nur für Amazon DocumentDB zu filtern.
- **`--source-identifier`**— Fakultativ. ID der Ereignisquelle, für die Ereignisse zurückgegeben werden. Wenn diese Option weggelassen wird, werden Ereignisse aus allen Quellen in die Ergebnisse einbezogen.
- **`--source-type`**— Optional, sofern nicht `--source-identifier` angegeben, dann erforderlich. Wenn `--source-identifier` angegeben wird, muss `--source-type` mit dem Typ der `--source-identifier` übereinstimmen. Die folgenden Werte sind zulässig:
  - `db-cluster`
  - `db-instance`
  - `db-parameter-group`
  - `db-security-group`
  - `db-cluster-snapshot`

Das folgende Beispiel listet alle Ihre Amazon DocumentDB DocumentDB-Ereignisse auf.

```
aws docdb describe-events --filters Name=engine,Values=docdb
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    "Events": [
        {
            "SourceArn": "arn:aws:rds:us-east-1:123SAMPLE012:db:sample-cluster-
instance3",
            "Message": "instance created",
            "SourceType": "db-instance",
            "Date": "2018-12-11T21:17:40.023Z",
            "SourceIdentifier": "sample-cluster-instance3",
            "EventCategories": [
                "creation"
            ]
```

```
        },
        {
            "SourceArn": "arn:aws:rds:us-
east-1:123SAMPLE012:db:docdb-2018-12-11-21-08-23",
            "Message": "instance shutdown",
            "SourceType": "db-instance",
            "Date": "2018-12-11T21:25:01.245Z",
            "SourceIdentifier": "docdb-2018-12-11-21-08-23",
            "EventCategories": [
                "availability"
            ]
        },
        {
            "SourceArn": "arn:aws:rds:us-
east-1:123SAMPLE012:db:docdb-2018-12-11-21-08-23",
            "Message": "instance restarted",
            "SourceType": "db-instance",
            "Date": "2018-12-11T21:25:11.441Z",
            "SourceIdentifier": "docdb-2018-12-11-21-08-23",
            "EventCategories": [
                "availability"
            ]
        }
    ]
}
```

Weitere Informationen finden Sie unter Prüfung von Amazon DocumentDB-Ereignissen.

Verwaltung von Amazon DocumentDB-Cluster-Parametergruppen

Sie können die Amazon DocumentDB-Engine-Konfiguration mithilfe von Parametern in einer Cluster-Parametergruppe verwalten. Eine Cluster-Parametergruppe ist eine Sammlung von Amazon DocumentDB-Konfigurationswerten, die die Verwaltung der Parameter Ihrer Amazon DocumentDB-Cluster erleichtern. Cluster-Parametergruppen dienen als Container für Engine-Konfigurationswerte, die auf jede alle Instances im Cluster angewendet werden.

In diesem Abschnitt wird beschrieben, wie Cluster-Parametergruppen erstellt, angezeigt und geändert werden. Es wird auch gezeigt, wie Sie ermitteln können, welche Cluster-Parametergruppe einem bestimmten Cluster zugeordnet ist.

Themen

- Beschreibung der Amazon DocumentDB-Cluster-Parametergruppen
- Erstellen von Amazon DocumentDB-Cluster-Parametergruppen
- Ändern der Amazon DocumentDB-Cluster-Parametergruppen
- Änderung von Amazon DocumentDB-Clustern zur Verwendung benutzerdefinierter Cluster-Parametergruppen
- Kopieren von Amazon DocumentDB-Cluster-Parametergruppen
- Amazon DocumentDB-Cluster-Parametergruppen werden zurückgesetzt
- Löschen von Amazon DocumentDB-Cluster-Parametergruppen
- Referenz zu den Amazon DocumentDB-Clusterparametern

Beschreibung der Amazon DocumentDB-Cluster-Parametergruppen

Eine `default` Cluster-Parametergruppe wird automatisch erstellt, wenn Sie den ersten Amazon DocumentDB-Cluster in einer neuen Engine erstellen AWS-Region oder eine neue Engine verwenden. Nachfolgende Cluster, die in derselben Region erstellt wurden und dieselbe Engine-Version haben, werden mit der `default` Cluster-Parametergruppe erstellt.

Themen

- Beschreibung der Details einer Amazon DocumentDB-Cluster-Parametergruppe
- Bestimmung der Parametergruppe eines Amazon DocumentDB-Clusters
- Ermitteln von Clustern und Instances, die einer Amazon DocumentDB-Cluster-Parametergruppe zugeordnet sind

Beschreibung der Details einer Amazon DocumentDB-Cluster-Parametergruppe

Um die Details einer bestimmten Cluster-Parametergruppe zu beschreiben, führen Sie die folgenden Schritte mit dem AWS-Managementkonsole oder dem AWS Command Line Interface (AWS CLI) aus.

Using the AWS-Managementkonsole

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Wählen Sie im Navigationsbereich Parameter groups (Parametergruppen) aus.

> **Tip**
>
> Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Wählen Sie im Bereich Cluster parameter groups (Cluster-Parametergruppen) den Namen der Parametergruppe aus, deren Details Sie anzeigen möchten.
4. Auf der anschließend angezeigten Seite werden die Parameter der Parametergruppe, die letzte Aktivität und Tags angezeigt.

   - In Cluster parameters (Cluster-Parameter) werden Ihnen der Name, der aktuelle Wert, die zulässigen Werte, der Anwendungstyp, der Datentyp und die Beschreibung des Parameters angezeigt. Sie können auch erkennen, ob der Parameter geändert werden kann. Sie können einzelne Parameter ändern, indem Sie den Parameter auswählen und dann im Abschnitt Cluster-Parameter auf Bearbeiten klicken. Weitere Informationen finden Sie unter Änderung der Amazon DocumentDB-Cluster-Parameter.
   - In Recent events (Aktuelle Ereignisse) werden Ihnen die jeweils aktuellen Ereignisse für diese Parametergruppe angezeigt. Sie können diese Ereignisse über die Suchleiste in diesem Abschnitt filtern. Weitere Informationen finden Sie unter Amazon DocumentDB DocumentDB-Ereignisse verwalten.
   - Unter Tags werden die Tags angezeigt, die sich in dieser Cluster-Parametergruppe befinden. Sie können Tags hinzufügen oder entfernen, indem Sie im Abschnitt „Tags“ auf Bearbeiten klicken. Weitere Informationen finden Sie unter Taggen von Amazon DocumentDB DocumentDB-Ressourcen.

Using the AWS CLI

Sie können den `describe-db-cluster-parameter-groups` AWS CLI Befehl verwenden, um den Amazon-Ressourcennamen (ARN), die Familie, die Beschreibung und den Namen einer einzelnen Cluster-Parametergruppe oder aller Cluster-Parametergruppen, die Sie für Amazon DocumentDB haben, anzuzeigen. Sie können den `describe-db-cluster-parameters` AWS CLI Befehl auch verwenden, um die Parameter und ihre Details innerhalb einer einzelnen Cluster-Parametergruppe anzuzeigen.

- **`--describe-db-cluster-parameter-groups`**— Um eine Liste all Ihrer Cluster-Parametergruppen und ihrer Details zu sehen.
  - **`--db-cluster-parameter-group-name`**— Fakultativ. Der Name der Cluster-Parametergruppe, die beschrieben werden soll. Wenn dieser Parameter weggelassen wird, werden alle Cluster-Parametergruppen beschrieben.
- **`--describe-db-cluster-parameters`**— Um alle Parameter innerhalb einer Parametergruppe und ihre Werte aufzulisten.
  - **`--db-cluster-parameter-group name`** – Erforderlich. Der Name der Cluster-Parametergruppe, die beschrieben werden soll.

Example

Der folgende Code listet bis zu 100 Cluster-Parametergruppen zusammen mit ARN, Familie, Beschreibung und Name auf.

```
aws docdb describe-db-cluster-parameter-groups
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
          "DBClusterParameterGroups": [
              {
                  "DBClusterParameterGroupArn": "arn:aws:rds:us-
east-1:012345678912:cluster-pg:default.docdb4.0",
                  "DBParameterGroupFamily": "docdb4.0",
                  "Description": "Default cluster parameter group for docdb4.0",
                  "DBClusterParameterGroupName": "default.docdb4.0"
              },
              {
                  "DBClusterParameterGroupArn": "arn:aws:rds:us-
east-1:012345678912:cluster-pg:sample-parameter-group",
                  "DBParameterGroupFamily": "docdb4.0",
                  "Description": "Custom docdb4.0 parameter group",
                  "DBClusterParameterGroupName": "sample-parameter-group"
              }
          ]
}
```

Example

Der folgende Code listet den ARN, die Familie, die Beschreibung und den Namen für `sample-parameter-group` auf.

Für Linux, macOS oder Unix:

```
aws docdb describe-db-cluster-parameter-groups \
          --db-cluster-parameter-group-name sample-parameter-group
```

Für Windows:

```
aws docdb describe-db-cluster-parameter-groups ^
          --db-cluster-parameter-group-name sample-parameter-group
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
          "DBClusterParameterGroups": [
              {
                  "DBClusterParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:cluster-pg:sample-parameter-group",
                  "Description": "Custom docdb4.0 parameter group",
                  "DBParameterGroupFamily": "docdb4.0",
                  "DBClusterParameterGroupName": "sample-parameter-group"
              }
          ]
}
```

Example

Der folgende Code listet die Werte der Parameter in `sample-parameter-group` auf.

Für Linux, macOS oder Unix:

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

Für Windows:

```
aws docdb describe-db-cluster-parameters ^
    --db-cluster-parameter-group-name sample-parameter-group
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

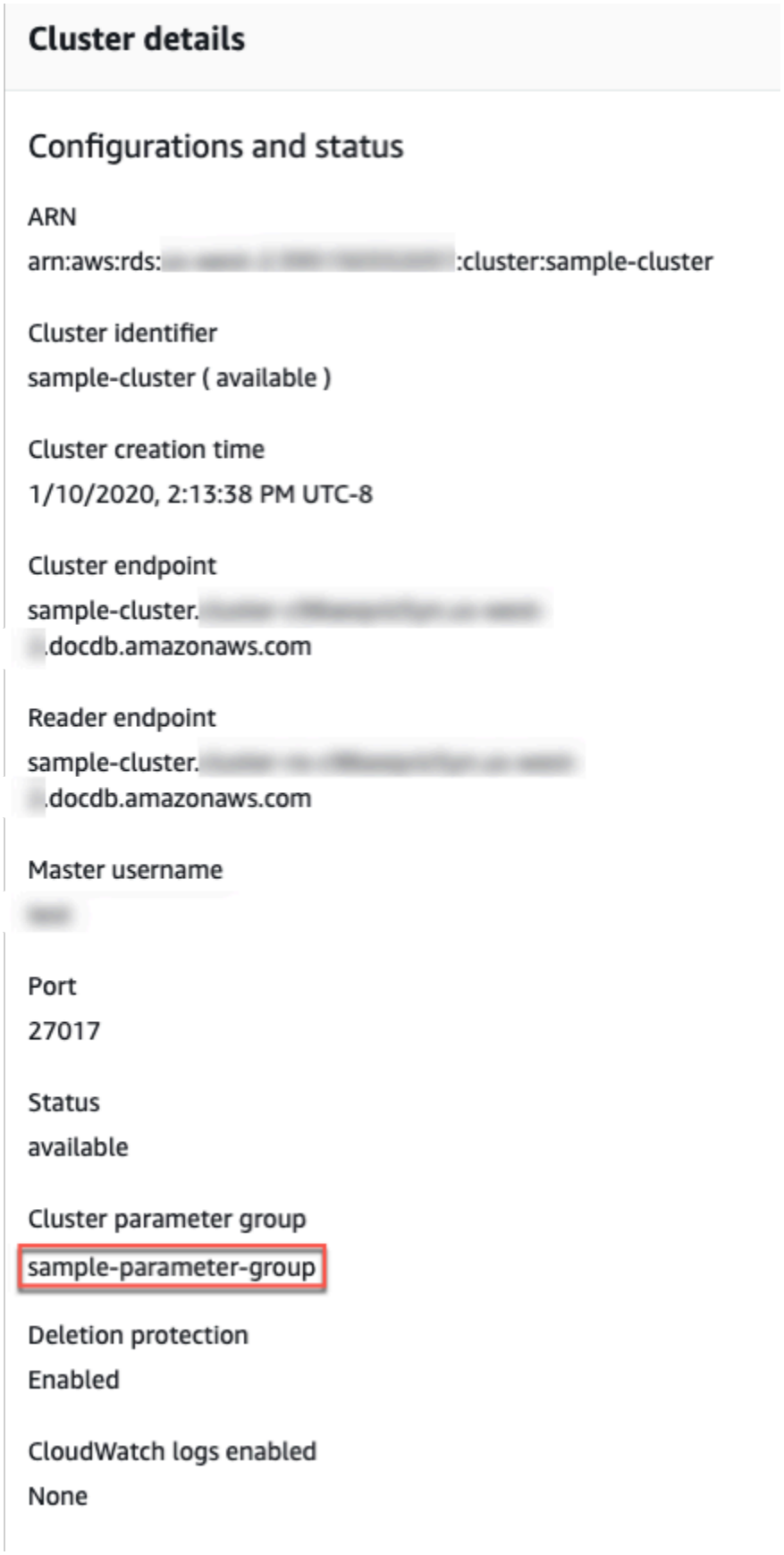
```
{
   "Parameters": [
         {
            "ParameterName": "audit_logs",
            "ParameterValue": "disabled",
            "Description": "Enables auditing on cluster.",
            "Source": "system",
            "ApplyType": "dynamic",
            "DataType": "string",
            "AllowedValues": "enabled,disabled",
            "IsModifiable": true,
            "ApplyMethod": "pending-reboot"
         },
         {
            "ParameterName": "change_stream_log_retention_duration",
            "ParameterValue": "17777",
            "Description": "Duration of time in seconds that the change stream log
 is retained and can be consumed.",
            "Source": "user",
            "ApplyType": "dynamic",
            "DataType": "integer",
            "AllowedValues": "3600-86400",
            "IsModifiable": true,
            "ApplyMethod": "pending-reboot"
         }
   ]
}
```

Bestimmung der Parametergruppe eines Amazon DocumentDB-Clusters

Um zu ermitteln, welche Parametergruppe einem bestimmten Cluster zugeordnet ist, führen Sie die folgenden Schritte mit AWS-Managementkonsole oder aus AWS CLI.

Using the AWS-Managementkonsole

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Wählen Sie im linken Navigationsbereich Cluster aus.
3. Wählen Sie aus der Liste der Cluster den Namen des Clusters aus, an dem Sie interessiert sind.
4. Auf der anschließend angezeigten Seite werden die Details des von Ihnen ausgewählten Clusters angezeigt. Scrollen Sie nach unten zu Cluster details (Clusterdetails). Sie finden den Namen der Parametergruppe unten in diesem Abschnitt unterhalb von Cluster parameter group (Cluster-Parametergruppe).


Cluster details
Configurations and status
ARN
arn:aws:rds:
:cluster:sample-cluster
Cluster identifier
sample-cluster (available)
Cluster creation time
1/10/2020, 2:13:38 PM UTC-8
Cluster endpoint
sample-cluster.
.docdb.amazonaws.com
Reader endpoint
sample-cluster.
.docdb.amazonaws.com
Master username
Port
27017
Status
available
Cluster parameter group
sample-parameter-group
Deletion protection
Enabled
CloudWatch logs enabled
None

Using the AWS CLI

Der folgende AWS CLI Code bestimmt, welche Parametergruppe den Cluster `sample-cluster` steuert.

```
aws docdb describe-db-clusters \
      --db-cluster-identifier sample-cluster \
      --query 'DBClusters[*].[DBClusterIdentifier,DBClusterParameterGroup]'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
[
       [
           "sample-cluster",
           "sample-parameter-group"
       ]
]
```

Ermitteln von Clustern und Instances, die einer Amazon DocumentDB-Cluster-Parametergruppe zugeordnet sind

Um zu ermitteln, wie viele Cluster und Instances den einzelnen Parametergruppen zugeordnet sind, führen Sie die folgenden Schritte mit dem aus AWS CLI.

Using the AWS CLI

Der folgende AWS CLI Code bestimmt, wie vielen Clustern und Instances zugeordnet sind`sample-parameter-group`.

Für Linux, macOS oder Unix:

```
aws docdb describe-db-clusters \
      --query 'DBClusters[*].
[DBClusterParameterGroup,DBClusterIdentifier,DBClusterMembers[*].DBInstanceIdentifier]'
 \
      --output json | \
      jq -r 'group_by(.[0]) | map({
         parameter_group_name: .[0][0],
         total_clusters: length,
         total_instances: map(.[2] | length) | add // 0
```

```
      }) | .[]'
```

Für Windows:

```
aws docdb describe-db-clusters ^
      --query 'DBClusters[*].
[DBClusterParameterGroup,DBClusterIdentifier,DBClusterMembers[*].DBInstanceIdentifier]'
 ^
      --output json | ^
      jq -r 'group_by(.[0]) | map({
         parameter_group_name: .[0][0],
         total_clusters: length,
         total_instances: map(.[2] | length) | add // 0
      }) | .[]'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
   "parameter_group_name": "sample-parameter-group",
   "total_clusters": 50,
   "total_instances": 150
}
```

Erstellen von Amazon DocumentDB-Cluster-Parametergruppen

Standard-Cluster-Parametergruppen wie`default.docdb5.0`, oder`default.docdb4.0`, werden erstellt`default.docdb3.6`, wenn Sie einen Cluster mit einer neuen Engine-Version und in einer neuen erstellen. AWS-Region Nachfolgende Cluster, die in derselben Region und mit derselben Engine-Version erstellt wurden, erben die `default` Cluster-Parametergruppe. Nach der Erstellung können die `default` Parametergruppen nicht gelöscht oder umbenannt werden. Sie können das Engine-Verhalten von Cluster-Instances ändern, indem Sie eine benutzerdefinierte Parametergruppe mit bevorzugten Parameterwerten erstellen und sie an Ihren Amazon DocumentDB-Cluster anhängen.

Das folgende Verfahren führt Sie durch die Erstellung einer benutzerdefinierten Cluster-Parametergruppe. Anschließend können Sie die Parameter innerhalb dieser Parametergruppe ändern.

> **Note**
>
> Sie sollten nach der Erstellung einer Cluster-Parametergruppe mindestens 5 Minuten warten, bevor Sie diese Cluster-Parametergruppe verwenden. Dadurch kann Amazon DocumentDB die `create` Aktion vollständig abschließen, bevor die Cluster-Parametergruppe für einen neuen Cluster verwendet wird. Sie können die `describe-db-cluster-parameter-groups` AWS CLI Operation AWS-Managementkonsole or the verwenden, um zu überprüfen, ob Ihre Cluster-Parametergruppe erstellt wurde. Weitere Informationen finden Sie unter Beschreibung der Amazon DocumentDB-Cluster-Parametergruppen.

Using the AWS-Managementkonsole

So erstellen Sie eine Cluster-Parametergruppe

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Wählen Sie im Navigationsbereich Parameter groups (Parametergruppen) aus.

   > **Tip**
   >
   > Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Wählen Sie im Bereich Cluster parameter groups (Cluster-Parametergruppen) die Option Create (Erstellen) aus.
4. Geben Sie im Bereich Create cluster parameter group (Cluster-Parametergruppe erstellen) Folgendes ein:

   a. Neuer Cluster-Parametergruppenname — Geben Sie einen Namen für die Cluster-Parametergruppe ein. Beispiel, `sample-parameter-group`. Für Cluster-Parametergruppen gelten die folgenden Benennungseinschränkungen:

      - Die Länge muss [1 bis 255] alphanumerische Zeichen betragen.
      - Muss mit einem Buchstaben beginnen.
      - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.

b. Familie — Wählen Sie eine DocumentDB-Version, die Sie für Ihren Cluster verwenden möchten.

c. Beschreibung — Geben Sie eine Beschreibung für diese Cluster-Parametergruppe ein.

5. Um die Cluster-Parametergruppe zu erstellen, wählen Sie Erstellen aus. Um die Operation abzubrechen, wählen Sie Abbrechen aus.

6. Nach der Auswahl von Create (Erstellen) wird oben auf der Seite der folgende Text angezeigt, um zu bestätigen, dass die Cluster-Parametergruppe erfolgreich erstellt wurde:

```
Successfully created cluster parameter group 'sample-parameter-group'.
```

Using the AWS CLI

Um eine neue Cluster-Parametergruppe für Amazon DocumentDB 4.0-Cluster zu erstellen, verwenden Sie den AWS CLI `create-db-cluster-parameter-group` Vorgang mit den folgenden Parametern:

- **`--db-cluster-parameter-group-name`**— Der Name der benutzerdefinierten Cluster-Parametergruppe. Beispiel, `sample-parameter-group`.
- **`--db-cluster-parameter-group-family`**— Die Cluster-Parametergruppenfamilie, die als Vorlage für die benutzerdefinierte Cluster-Parametergruppe verwendet wird.
- **`--description`**— Die vom Benutzer bereitgestellte Beschreibung für diese Cluster-Parametergruppe. Im folgenden Beispiel wird "`Custom docdb4.0 parameter group`“ verwendet.

Für Linux, macOS oder Unix:

Example

```
aws docdb create-db-cluster-parameter-group \
 --db-cluster-parameter-group-name sample-parameter-group \
 --db-parameter-group-family docdb5.0 \
 --description "Custom docdb5.0 parameter group"
```

Für Windows:

```
aws docdb create-db-cluster-parameter-group ^
 --db-cluster-parameter-group-name sample-parameter-group ^
```

```
  --db-parameter-group-family docdb5.0 ^
  --description "Custom docdb5.0 parameter group"
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "sample-parameter-group",
        "DBParameterGroupFamily": "docdb5.0",
        "Description": "Custom docdb4.0 parameter group",
        "DBClusterParameterGroupArn": "sample-parameter-group-arn"
    }
}
```

Ändern der Amazon DocumentDB-Cluster-Parametergruppen

In diesem Abschnitt wird erklärt, wie eine benutzerdefinierte Amazon DocumentDB-Parametergruppe geändert wird. In Amazon DocumentDB können Sie eine `default` Cluster-Parametergruppe nicht ändern, die erstellt wird, wenn Sie zum ersten Mal einen Cluster mit einer neuen Engine-Version in einer neuen Region erstellen. Wenn Ihr Amazon DocumentDB-Cluster die Standard-Cluster-Parametergruppe verwendet und Sie einen Wert darin ändern möchten, müssen Sie zuerst eine neue Parametergruppe erstellen oder eine vorhandene Parametergruppe kopieren, sie ändern und dann die geänderte Parametergruppe auf Ihren Cluster anwenden.

Führen Sie die folgenden Schritte aus, um eine benutzerdefinierte Cluster-Parametergruppe zu ändern. Es kann eine Weile dauern, bis Änderungsaktionen übernommen werden. Warten Sie, bis die geänderte Cluster-Parametergruppe verfügbar ist, bevor Sie sie an Ihren Cluster anhängen. Sie können den AWS CLI `describe-db-cluster-parameters` Vorgang AWS-Managementkonsole oder verwenden, um zu überprüfen, ob Ihre Cluster-Parametergruppe geändert wurde. Weitere Informationen finden Sie unter Beschreibung der Cluster-Parametergruppen.

Using the AWS-Managementkonsole

Folgen Sie diesen Schritten, um eine benutzerdefinierte Amazon DocumentDB-Parametergruppe zu ändern. Sie können eine `default`-Parametergruppe nicht ändern. Wenn Sie einen Wert in der `default`-Parametergruppe ändern möchten, können Sie die standardmäßige Cluster-Parametergruppe kopieren, sie ändern und die geänderte Parametergruppe anschließend auf Ihren Cluster anwenden. Weitere Hinweise zum Anwenden von Parametergruppen auf Ihren Cluster finden Sie unter Ändern eines Amazon DocumentDB-Clusters.

So ändern Sie eine benutzerdefinierte Cluster-Parametergruppe

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Wählen Sie im Navigationsbereich auf der linken Seite der Konsole Parameter groups (Parametergruppen) aus. Wählen Sie in der Liste der Parametergruppen den Namen der Parametergruppe, die Sie ändern möchten.

   > **Tip**
   >
   > Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Gehen Sie bei jedem Parameter in der Parametergruppe, die Sie ändern möchten, wie folgt vor:

   a. Suchen Sie den Parameter, den Sie ändern möchten, und überprüfen Sie, ob er modifizierbar ist, indem Sie überprüfen, ob er in der `true` Spalte Modifizierbar aufgeführt ist.

   b. Wenn der Parameter geändert werden kann, wählen Sie ihn aus und wählen oben rechts auf der Konsolenseite Edit (Bearbeiten) aus.

   c. Nehmen Sie im Dialogfeld Modify **<parameter-name>** die gewünschten Änderungen vor. Wählen Sie anschließend Modify cluster parameter (Cluster-Parameter ändern) aus. Um die Änderungen zu verwerfen, wählen Sie Cancel (Abbrechen) aus.

Using the AWS CLI

Sie können den `ParameterValue` oder einen beliebigen modifizierbaren Parameter in `ApplyMethod` einer benutzerdefinierten Amazon DocumentDB-Cluster-Parametergruppe ändern, indem Sie den verwenden. `Description` AWS CLI Sie können eine standardmäßige Cluster-Parametergruppe nicht direkt ändern.

Um die Parameter einer benutzerdefinierten Cluster-Parametergruppe zu ändern, verwenden Sie die Operation `modify-db-cluster-parameter-group` mit den folgenden Parametern.

- **`--db-cluster-parameter-group-name`** – Erforderlich. Der Name der Cluster-Parametergruppe, die Sie ändern.

- **`--parameters`** – Erforderlich. Die Parameter, die von Ihnen geändert werden. Eine Liste der Parameter, die für alle Instances in einem Amazon DocumentDB-Cluster gelten, finden Sie im. Referenz zu den Amazon DocumentDB-Clusterparametern Jeder Parametereintrag muss Folgendes enthalten:
  - **`ParameterName`**— Der Name des Parameters, den Sie ändern.
  - **`ParameterValue`**— Der neue Wert für diesen Parameter.
  - **`ApplyMethod`**— Wie Sie Änderungen an diesem Parameter anwenden möchten. Zugelassene Werte sind `immediate` und `pending-reboot`.

    > **Note**
    >
    > Parameter mit dem `ApplyType` von `static` müssen über einen `ApplyMethod` von `pending-reboot` verfügen.

Example- Ändern eines Parameterwerts

In diesem Beispiel listen Sie die Parameterwerte von `sample-parameter-group` auf und ändern den `tls`-Parameter. Anschließend listen Sie nach einer Wartezeit von 5 Minuten die Parameterwerte von `sample-parameter-group` erneut auf, um die geänderten Parameterwerte anzuzeigen.

1. Listen Sie die Parameter von `sample-parameter-group` und ihre Werte auf.

   Für Linux, macOS oder Unix:

   ```
   aws docdb describe-db-cluster-parameters \
          --db-cluster-parameter-group-name sample-parameter-group
   ```

   Für Windows:

   ```
   aws docdb describe-db-cluster-parameters ^
          --db-cluster-parameter-group-name sample-parameter-group
   ```

   Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

   ```
   {
          "Parameters": [
   ```

```
            {
                "Source": "system",
                "ApplyType": "static",
                "AllowedValues": "disabled,enabled",
                "ParameterValue": "enabled",
                "ApplyMethod": "pending-reboot",
                "DataType": "string",
                "ParameterName": "tls",
                "IsModifiable": true,
                "Description": "Config to enable/disable TLS"
            },
            {
                "Source": "user",
                "ApplyType": "dynamic",
                "AllowedValues": "disabled,enabled",
                "ParameterValue": "enabled",
                "ApplyMethod": "pending-reboot",
                "DataType": "string",
                "ParameterName": "ttl_monitor",
                "IsModifiable": true,
                "Description": "Enables TTL Monitoring"
            }
        ]
}
```

2. Ändern Sie den Parameter `tls`, sodass der Wert `disabled` ist.

   Sie können den `ApplyMethod` nicht ändern, da der `ApplyType` `static` ist.

   Für Linux, macOS oder Unix:

```
aws docdb modify-db-cluster-parameter-group \
       --db-cluster-parameter-group-name sample-parameter-group \
       --parameters
 "ParameterName"=tls,"ParameterValue"=disabled,"ApplyMethod"=pending-reboot
```

   Für Windows:

```
aws docdb modify-db-cluster-parameter-group ^
       --db-cluster-parameter-group-name sample-parameter-group ^
       --parameters
 "ParameterName"=tls,"ParameterValue"=disabled,"ApplyMethod"=pending-reboot
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
       "DBClusterParameterGroupName": "sample-parameter-group"
   }
```

3. Warten Sie mindestens 5 Minuten.
4. Listen Sie die Parameterwerte von `sample-parameter-group` auf, um zu überprüfen, ob der Parameter `tls` geändert wurde.

   Für Linux, macOS oder Unix:

```
aws docdb describe-db-cluster-parameters \
       --db-cluster-parameter-group-name sample-parameter-group
```

   Für Windows:

```
aws docdb describe-db-cluster-parameters ^
       --db-cluster-parameter-group-name sample-parameter-group
```

   Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
       "Parameters": [
           {
               "ParameterValue": "false",
               "ParameterName": "enable_audit_logs",
               "ApplyType": "dynamic",
               "DataType": "string",
               "Description": "Enables auditing on cluster.",
               "AllowedValues": "true,false",
               "Source": "system",
               "IsModifiable": true,
               "ApplyMethod": "pending-reboot"
           },
           {
               "ParameterValue": "disabled",
               "ParameterName": "tls",
               "ApplyType": "static",
               "DataType": "string",
```

```
                "Description": "Config to enable/disable TLS",
                "AllowedValues": "disabled,enabled",
                "Source": "system",
                "IsModifiable": true,
                "ApplyMethod": "pending-reboot"
            }
        ]
}
```

Änderung von Amazon DocumentDB-Clustern zur Verwendung benutzerdefinierter Cluster-Parametergruppen

Wenn Sie einen Amazon DocumentDB-Cluster erstellen, wird automatisch eine `default.docdb4.0` Parametergruppe für diesen Cluster erstellt. Sie können die Cluster-Parametergruppe `default` nicht ändern. Stattdessen können Sie Ihren Amazon DocumentDB-Cluster so ändern, dass er ihm eine neue benutzerdefinierte Parametergruppe zuordnet.

In diesem Abschnitt wird erklärt, wie Sie einen vorhandenen Amazon DocumentDB-Cluster so ändern, dass er eine benutzerdefinierte Cluster-Parametergruppe mit AWS-Managementkonsole und AWS Command Line Interface ()AWS CLI verwendet.

Using the AWS-Managementkonsole

Um einen Amazon DocumentDB-Cluster so zu ändern, dass er eine neue, nicht standardmäßige Cluster-Parametergruppe verwendet

1. Bevor Sie beginnen, stellen Sie sicher, dass Sie einen Amazon DocumentDB-Cluster und eine Cluster-Parametergruppe erstellt haben. Weitere Anweisungen finden Sie unter Erstellen eines Amazon DocumentDB-Clusters und Erstellen von Amazon DocumentDB-Cluster-Parametergruppen.
2. Nachdem Sie Ihre Cluster-Parametergruppe erstellt haben, öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb Wählen Sie im Navigationsbereich Clusters (Cluster) aus, um die neue Parametergruppe einem Cluster hinzuzufügen.
3. Wählen Sie den Cluster aus, dem Sie die Parametergruppe zuordnen möchten. Wählen Sie Actions (Aktionen= und anschließend Modify (Ändern) aus, um den Cluster zu ändern.
4. Wählen Sie unter Cluster options (Clusteroptionen) die neue Parametergruppe aus, der Sie den Cluster zuordnen möchten.

5. Wählen Sie Continue (Weiter) aus, um eine Übersicht Ihrer Änderungen anzuzeigen.
6. Nachdem Sie Ihre Änderungen überprüft haben, können Sie diese sofort oder während des nächsten Wartungsfensters unter Scheduling of modifications (Planen von Änderungen) anwenden.
7. Wählen Sie Modify cluster (Cluster ändern) aus, um den Cluster mit der neuen Parametergruppe zu aktualisieren.

Using the AWS CLI

Bevor Sie beginnen, stellen Sie sicher, dass Sie einen Amazon DocumentDB-Cluster und eine Cluster-Parametergruppe erstellt haben. Sie können mithilfe der Operation einen Amazon DocumentDB-Cluster erstellen. AWS CLI `create-db-cluster` Sie können mithilfe der AWS CLI`create-db-cluster-parameter-group` Operation eine Cluster-Parametergruppe erstellen.

Um Ihre neue Cluster-Parametergruppe zu Ihrem Cluster hinzuzufügen, verwenden Sie den AWS CLI `modify-db-cluster` Vorgang mit den folgenden Parametern.

- --db-cluster-identifier — Der Name Ihres Clusters (zum Beispiel). `sample-cluster`
- --db-cluster-parameter-group-name — Der Name der Parametergruppe, der Sie Ihren Cluster zuordnen möchten (z. B.). `sample-parameter-group`

Example

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --db-cluster-parameter-group-name sample-parameter-group
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
"DBCluster": {
    "AvailabilityZones": [
            "us-west-2c",
            "us-west-2b",
            "us-west-2a"
    ],
    "BackupRetentionPeriod": 1,
    "DBClusterIdentifier": "sample-cluster",
    "DBClusterParameterGroup": "sample-parameter-group",
```

```
    "DBSubnetGroup": "default",
   ...
}
```

Kopieren von Amazon DocumentDB-Cluster-Parametergruppen

Sie können eine Kopie einer Cluster-Parametergruppe in Amazon DocumentDB mit AWS-Managementkonsole oder AWS Command Line Interface ()AWS CLI erstellen.

Using the AWS-Managementkonsole

Das folgende Verfahren führt Sie durch die Erstellung einer neuen Cluster-Parametergruppe, indem Sie eine vorhandene Cluster-Parametergruppe kopieren.

So kopieren Sie eine Cluster-Parametergruppe

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Wählen Sie im Navigationsbereich Parameter groups (Parametergruppen) aus.
3. Wählen Sie im Bereich Cluster parameter groups (Cluster-Parametergruppen) den Namen der Cluster-Parametergruppe aus, die Sie kopieren möchten.
4. Wählen Sie Actions (Aktionen) und anschließend Copy (Kopieren) aus, um diese Parametergruppe zu kopieren.
5. Geben Sie in Copy options (Kopieroptionen) einen Namen und eine Beschreibung für die neue Cluster-Parametergruppe ein. Wählen Sie anschließend Copy (Kopieren) aus, um Ihre Änderungen zu speichern.

Using the AWS CLI

Um eine Kopie einer Cluster-Parametergruppe zu erstellen, verwenden Sie die `copy-db-cluster-parameter-group`-Operation mit den folgenden Parametern.

- **`--source-db-cluster-parameter-group-identifier`** – Erforderlich. Der Name oder Amazon-Ressourcenname (ARN) der Cluster-Parametergruppe, von der Sie eine Kopie erstellen möchten.

  Wenn die Quell- und Zielcluster-Parametergruppen identisch sind AWS-Region, kann der Bezeichner entweder ein Name oder ein ARN sein.

Wenn die Quell- und Zielcluster-Parametergruppen unterschiedlich sind AWS-Regionen, muss der Bezeichner ein ARN sein.

- **`--target-db-cluster-parameter-group-identifier`** – Erforderlich. Der Name oder ARN der Kopie der Cluster-Parametergruppe.

  Einschränkungen:

  - Kann nicht Null, leer oder negativ sein.
  - Muss 1—255 Buchstaben, Zahlen oder Bindestriche enthalten.
  - Muss mit einem Buchstaben beginnen.
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.
- **`--target-db-cluster-parameter-group-description`** – Erforderlich. Eine vom Benutzer angegebene Beschreibung für die Kopie der Cluster-Parametergruppe.

Example

Der folgende Code erstellt eine Kopie von `sample-parameter-group` und benennt die Kopie `sample-parameter-group-copy`.

Für Linux, macOS oder Unix:

```
aws docdb copy-db-cluster-parameter-group \
    --source-db-cluster-parameter-group-identifier sample-parameter-group \
    --target-db-cluster-parameter-group-identifier sample-parameter-group-copy \
    --target-db-cluster-parameter-group-description "Copy of sample-parameter-group"
```

Für Windows:

```
aws docdb copy-db-cluster-parameter-group ^
    --source-db-cluster-parameter-group-identifier sample-parameter-group ^
    --target-db-cluster-parameter-group-identifier sample-parameter-group-copy ^
    --target-db-cluster-parameter-group-description "Copy of sample-parameter-group"
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    "DBClusterParameterGroup": {
```

```
        "DBClusterParameterGroupArn": "arn:aws:rds:us-east-1:123456789012:cluster-
pg:sample-parameter-group-copy",
        "DBClusterParameterGroupName": "sample-parameter-group-copy",
        "DBParameterGroupFamily": "docdb4.0",
        "Description": "Copy of sample-parameter-group"
    }
}
```

Amazon DocumentDB-Cluster-Parametergruppen werden zurückgesetzt

Sie können einige oder alle Parameterwerte einer Amazon DocumentDB-Cluster-Parametergruppe auf ihre Standardwerte zurücksetzen, indem Sie die Cluster-Parametergruppe mithilfe von AWS-Managementkonsole oder AWS Command Line Interface (AWS CLI) zurücksetzen.

Using the AWS-Managementkonsole

Führen Sie die folgenden Schritt aus, um einige oder alle Parameterwerte einer Cluster-Parametergruppe auf die Standardwerte zurückzusetzen.

So setzen Sie die Parameterwerte einer Cluster-Parametergruppe zurück

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Wählen Sie im Navigationsbereich auf der linken Seite der Konsole Parameter groups (Parametergruppen) aus.
3. Wählen Sie im Bereich Cluster parameter groups (Cluster-Parametergruppen) den Namen der Cluster-Parametergruppe aus, die Sie zurücksetzen möchten.
4. Wählen Sie Actions (Aktionen) und anschließend Reset (Zurücksetzen) aus, um diese Parametergruppe zurückzusetzen.
5. Bestätigen Sie auf der Seite Cluster parameter group reset confirmation (Zurücksetzen der Cluster-Parametergruppe bestätigen), dass Sie alle Cluster-Parameter für diese Parametergruppe auf die Standardwerte zurücksetzen möchten. Wählen Sie anschließend Reset (Zurücksetzen) aus, um die Parametergruppe zurückzusetzen. Sie können auch Cancel (Abbrechen) auswählen, um die Änderungen zu verwerfen.

Using the AWS CLI

Um einige oder alle Parameterwerte einer Cluster-Parametergruppe auf ihre Standardwerte zurückzusetzen, verwenden Sie die `reset-db-cluster-parameter-group`-Operation mit den folgenden Parametern.

- **`--db-cluster-parameter-group-name`** – Erforderlich. Der Name der zurückzusetzenden Cluster-Parametergruppe.
- **`--parameters`**— Fakultativ. Eine Liste von `ParameterName` und `ApplyMethod` in der Cluster-Parametergruppe, die auf ihre Standardwerte zurücksetzt werden sollen. Statische Parameter müssen auf `pending-reboot` gesetzt sein, damit sie beim nächsten Neustart der Instance oder `reboot-db-instance`-Anforderung wirksam werden. Sie müssen `reboot-db-instance` für alle Instances in Ihrem Cluster aufrufen, auf die der aktualisierte statische Parameter angewendet werden soll.

  Dieser Parameter und `--reset-all-parameters` schließen sich gegenseitig aus. Sie können jede der beiden Optionen verwenden, jedoch nicht beide.
- **`--reset-all-parameters`**oder **`--no-reset-all-parameters`** — Fakultativ. Gibt an, ob alle Parameter (`--reset-all-parameters`) oder nur einige der Parameter (`--no-reset-all-parameters`) auf ihre Standardwerte zurückgesetzt werden sollen. Der `--reset-all-parameters`-Parameter und `--parameters` schließen sich gegenseitig aus. Sie können jede der beiden Optionen verwenden, jedoch nicht beide.

  Wenn Sie die gesamte Gruppe zurücksetzen, werden dynamische Parameter sofort aktualisiert. Statische Parameter werden auf `pending-reboot` festgelegt, damit sie beim nächsten Neustart der Instance oder bei der nächsten `reboot-db-instance`-Anforderung wirksam werden. Sie müssen `reboot-db-instance` für alle Instances in Ihrem Cluster aufrufen, auf die der aktualisierte statische Parameter angewendet werden soll.

Example

Beispiel 1: Zurücksetzen aller Parameter auf ihre Standardwerte

Der folgende Code setzt alle Parameter in der Cluster-Parametergruppe `sample-parameter-group` auf ihre Standardwerte zurück.

Für Linux, macOS oder Unix:

```
aws docdb reset-db-cluster-parameter-group \
```

```
        --db-cluster-parameter-group-name sample-parameter-group \
        --reset-all-parameters
```

Für Windows:

```
aws docdb reset-db-cluster-parameter-group ^
        --db-cluster-parameter-group-name sample-parameter-group ^
        --reset-all-parameters
```

Beispiel 2: Zurücksetzen von angegebenen Parametern auf ihre Standardwerte

Der folgende Code setzt den `tls`-Parameter in der Cluster-Parametergruppe `sample-parameter-group` auf seinen Standardwert zurück.

Für Linux, macOS oder Unix:

```
aws docdb reset-db-cluster-parameter-group \
        --db-cluster-parameter-group-name sample-parameter-group \
        --no-reset-all-parameters \
        --parameters ParameterName=tls,ApplyMethod=pending-reboot
```

Für Windows:

```
aws docdb reset-db-cluster-parameter-group ^
        --db-cluster-parameter-group-name sample-parameter-group ^
        --no-reset-all-parameters ^
        --parameters ParameterName=tls,ApplyMethod=pending-reboot
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
        "DBClusterParameterGroupName": "sample-parameter-group"
   }
```

Neustarten einer Cluster-Instance

Bevor ein statischer Parameterwert geändert wird, muss die Cluster-Instance neu gestartet werden. Starten Sie alle Instances in Ihrem Cluster neu, auf die der aktualisierte statische Parameter angewendet werden soll.

Für Linux, macOS oder Unix:

```
aws docdb reboot-db-instance \
       --db-instance-identifier sample-cluster-instance
```

Für Windows:

```
aws docdb reboot-db-instance ^
       --db-instance-identifier sample-cluster-instance
```

Löschen von Amazon DocumentDB-Cluster-Parametergruppen

Sie können eine benutzerdefinierte Amazon DocumentDB-Cluster-Parametergruppe mit AWS-Managementkonsole oder AWS Command Line Interface ()AWS CLI löschen. Sie können die Cluster-Parametergruppe `default.docdb4.0` nicht löschen.

Using the AWS-Managementkonsole

So löschen Sie eine Cluster-Parametergruppe

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Wählen Sie im Navigationsbereich Parameter groups (Parametergruppen) aus.

   > **Tip**
   >
   > Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Wählen Sie im Bereich Parametergruppen das Optionsfeld links neben der Cluster-Parametergruppe aus, die Sie löschen möchten.
4. Wählen Sie Aktionen und anschließend Löschen aus.
5. Wählen Sie im Bestätigungsbereich Delete (Löschen) die Option Delete (Löschen) aus, um die Cluster-Parametergruppe zu löschen. Um die Cluster-Parametergruppe beizubehalten, wählen Sie Cancel (Abbrechen) aus.

Using the AWS CLI

Um eine Cluster-Parametergruppe zu löschen, verwenden Sie die `delete-db-cluster-parameter-group`-Operation mit dem folgenden Parameter.

- **`--db-cluster-parameter-group-name`** – Erforderlich. Der Name der zu löschenden Cluster-Parametergruppe. Dabei muss es sich um eine vorhandene Cluster-Parametergruppe handeln. Sie können die Cluster-Parametergruppe *`default.docdb4.0`* nicht löschen.

Example- Löschen einer Cluster-Parametergruppe

Das folgende Beispiel führt Sie durch die drei Schritte zum Löschen einer Cluster-Parametergruppe:

1. Ermitteln des Namens der Cluster-Parametergruppe, die Sie löschen möchten.
2. Löschen der angegebene Cluster-Parametergruppe.
3. Überprüfen, ob die Cluster-Parametergruppe gelöscht wurde.

1. Suchen Sie den Namen der Cluster-Parametergruppe, die Sie löschen möchten.

Der folgende Code listet die Namen aller Cluster-Parametergruppen auf.

Für Linux, macOS oder Unix:

```
aws docdb describe-db-cluster-parameter-groups \
       --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

Für Windows:

```
aws docdb describe-db-cluster-parameter-groups ^
       --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

Die Ausgabe der vorherigen Operation ist eine Liste der Namen der Cluster-Parametergruppen ähnlich der folgenden (JSON-Format).

```
[
       [
           "default.docdb4.0"
       ],
       [
```

```
            "sample-parameter-group"
        ],
        [
            "sample-parameter-group-copy"
        ]
    ]
```

2. Löschen Sie eine angegebene Cluster-Parametergruppe.

Der folgenden Code löscht die Cluster-Parametergruppe `sample-parameter-group-copy`.

Für Linux, macOS oder Unix:

```
aws docdb delete-db-cluster-parameter-group \
       --db-cluster-parameter-group-name sample-parameter-group-copy
```

Für Windows:

```
aws docdb delete-db-cluster-parameter-group ^
       --db-cluster-parameter-group-name sample-parameter-group-copy
```

Diese Operation erzeugt keine Ausgabe.

3. Überprüfen Sie, ob die angegebene Cluster-Parametergruppe gelöscht wurde.

Der folgende Code listet die Namen aller verbleibenden Cluster-Parametergruppen auf.

Für Linux, macOS oder Unix:

```
aws docdb describe-db-cluster-parameter-groups \
       --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

Für Windows:

```
aws docdb describe-db-cluster-parameter-groups ^
       --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

Die Ausgabe der obigen Operation ist eine Liste der Cluster-Parametergruppen, die der folgenden ähnelt (JSON-Format). Die Cluster-Parametergruppe, die Sie gerade gelöscht haben, sollte nicht in der Liste enthalten sein.

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
[
       [
           "default.docdb4.0"
       ],
       [
           "sample-parameter-group"
       ]
   ]
```

Referenz zu den Amazon DocumentDB-Clusterparametern

Wenn Sie einen dynamischen Parameter ändern und die Cluster-Parametergruppe speichern, wird diese Änderung sofort übernommen, ungeachtet der Einstellung von Apply immediately (Sofort anwenden). Wenn Sie einen statischen Parameter ändern und eine Cluster-Parametergruppe speichern, wird die Änderung des Parameters nach einem manuellen Neustart der Instance angewendet. Sie können eine Instance mithilfe der Amazon DocumentDB-Konsole oder durch einen expliziten Aufruf neu starten. `reboot-db-instance`

Die folgende Tabelle zeigt die Parameter, die für alle Instances in einem Amazon DocumentDB-Cluster gelten.

Amazon DocumentDB-Parameter auf Cluster-Ebene

Parameter	Standardwert	Zulässige Werte	Anpassbar	Typ anwenden	Datentyp	Description
`audit_logs`	disabled	aktiviert, deaktiviert, ddl, dml_read, dml_write, all, none	Ja	Dynamisch	Zeichenfolge	Definiert, ob Amazon-Auditprotokolle aktiviert sind. CloudWatch • **`enabled`**—CloudWatch

Parameter	Standardwert	Zulässige Werte	Anpassbar	Typ anwenden	Datentyp	Description
						Auditprotokolle sind aktiviert. • **disabled**— CloudWatch Auditprotokolle sind deaktiviert. • **ddl**— Die Überwachung von DDL-Ereignissen ist aktiviert. • **dml_read**— Die Überwachung von DML-Leseereignissen ist aktiviert. • **dml_write** — Die Überwachu

Parameter	Standardwert	Zulässige Werte	Anpassbar	Typ anwenden	Datentyp	Description
						ng von DML-Schreibereignissen ist aktiviert. • **`all`**— Die Überwachung aller Datenbankereignisse ist aktiviert. • **`none`**— Die Überwachung ist deaktiviert.

Parameter	Standardwert	Zulässige Werte	Anpassbar	Typ anwenden	Datentyp	Description
`change_stream_log_retention_duration`	10800	3600-604800	Ja	Dynamisch	Ganzzahl	Definiert den Zeitraum (in Sekunden), den das Change Stream-Protokoll aufbewahrt und genutzt werden kann.

Parameter	Standardwert	Zulässige Werte	Anpassbar	Typ anwenden	Datentyp	Description
`default_collection_compression`	disabled	aktiviert, deaktiviert (Amazon DocumentDB 5.0)/zstd, lz4, none (Amazon) DocumentDB8.0	Ja	Dynamisch	Zeichenfolge	Definiert die Standardkomprimierungseinstellung für neue Sammlungen in einem Cluster • **enabled**— Die Lz4-Komprimierung ist in Amazon DocumentDB 5.0 standardmäßig aktiviert. Die zstd-Komprimierung ist in Amazon DocumentDB 8.0 standardmäßig aktiviert. • **disabled, none**—

Parameter	Standardwert	Zulässige Werte	Anpassbar	Typ anwenden	Datentyp	Description
						Die Komprimierung ist standardmäßig deaktiviert (in Amazon DocumentDB 5.0 deaktiviert, in Amazon DocumentDB 8.0 nicht).

Parameter	Standardwert	Zulässige Werte	Anpassbar	Typ anwenden	Datentyp	Description
`profiler`	disabled	aktiviert, deaktiviert	Ja	Dynamisch	Zeichenfolge	Aktiviert die Profilerstellung für langsame Operationen. • **`enabled`**— Vorgänge, die länger als ein vom Kunden definierter Schwellenwert dauern (z. B. 100 ms), werden in Amazon Logs protokolliert. CloudWatch • **`disabled`**— Langsame Vorgänge werden

Parameter	Standardwert	Zulässige Werte	Anpassbar	Typ anwenden	Datentyp	Description
						nicht in Logs protokolliert. CloudWatch
`profiler_sampling_rate`	1,0	0,0 – 1,0	Ja	Dynamisch	Gleitkommazahl	Definiert die Sampling-Rate für protokollierte Operationen.

Parameter	Standardwert	Zulässige Werte	Anpassbar	Typ anwenden	Datentyp	Description
`profiler_threshold_ms`	100	50 – 2147483646	Ja	Dynamisch	Ganzzahl	Definiert den Schwellenwert für `profiler`. • Alle Operationen, die größer als `profiler_threshold_ms` sind, werden in CloudWatch Logs protokolliert.

Parameter	Standardwert	Zulässige Werte	Anpassbar	Typ anwenden	Datentyp	Description
`planner_version`	3.0	1,0, 2,0, 3,0	Ja	Dynamisch	Gleitkommazahl	Definiert die Version des Abfrageplaners, die für Abfragen verwendet werden soll. • **`1.0`**— Wird in Amazon DocumentDB 5.0 unterstützt. • **`2.0`**— Verbesserungen beim Suchen und Aktualisieren von Operatoren. Wird in Amazon DocumentDB 5.0 unterstützt.

Parameter	Standardwert	Zulässige Werte	Anpassbar	Typ anwenden	Datentyp	Description
						• **`3.0`**— Verbesserungen beim Suchen, Aktualisieren und Aggregieren von Operatoren. Standard-Abfrageplaner in Amazon DocumentDB 8.0.

Parameter	Standardwert	Zulässige Werte	Anpassbar	Typ anwenden	Datentyp	Description
`tls`	aktiviert	aktiviert, deaktiviert, fips-140-3, tls1.2+, tls1.3+	Ja	Statisch	Zeichenfolge	Gibt an, ob TLS-Verbindungen (Transport Layer Security) erforderlich sind. • **`enabled`**— TLS-Verbindungen sind erforderlich, um eine Verbindung herzustellen. Für Amazon DocumentDB 5.0 und früher aktiviert dies TLS 1.0 bis 1.3, verwenden Sie jedoch TLS 1.2

Parameter	Standardwert	Zulässige Werte	Anpassbar	Typ anwenden	Datentyp	Description
						oder höher. Ab der Nebenversion 5.0.1, Amazon DocumentDB 8.0 und neuer wird auf Amazon DocumentDB 5.0 nur TLS 1.2 und höher unterstützt. • **`disabled`**— TLS-Verbindungen können nicht für die Verbindung verwendet werden. • **`fips-140-3`** — Für die

Parameter	Standardwert	Zulässige Werte	Anpassbar	Typ anwenden	Datentyp	Description
						Verbindung sind TLS-Verbindungen mit FIPS-Attributen (Federal Information Processing Standards) erforderlich. Der Cluster akzeptiert nur sichere Verbindungen gemäß FIPS-Publikation 140-3. Dies wird erst ab Amazon DocumentDB 5.0 (Engine-

Parameter	Standardwert	Zulässige Werte	Anpassbar	Typ anwenden	Datentyp	Description
						Version 3.0.3727) in den folgenden Regionen unterstützt: ca-central-1, us-west-2, us-east-1, us-east-2, us-gov-east-1, us-gov-west-1. • **tls1.2+**— TLS-Verbindungen mit TLS-Version 1.2 und höher sind erforderlich, um eine Verbindung herzustellen. Dies

Parameter	Standardwert	Zulässige Werte	Anpassbar	Typ anwenden	Datentyp	Description
						wird erst ab Amazon DocumentDB 4.0 (Engine-Version 2.0.10980) und Amazon DocumentDB 5.0 (Engine-Version 3.0.11051) unterstützt. • **`tls1.3+`**— TLS-Verbindungen mit TLS-Version 1.3 und höher sind erforderlich, um eine Verbindung herzustellen. Dies

Parameter	Standardwert	Zulässige Werte	Anpassbar	Typ anwenden	Datentyp	Description
						wird erst ab Amazon DocumentDB 4.0 (Engine-Version 2.0.10980) und Amazon DocumentDB 5.0 (Engine-Version 3.0.11051) unterstützt.

Parameter	Standardwert	Zulässige Werte	Anpassbar	Typ anwenden	Datentyp	Description
`ttl_monitor`	aktiviert	aktiviert, deaktiviert	Ja	Dynamisch	Zeichenfolge	Gibt an, ob TTL-Überwachung (Time to Live) für den Cluster aktiviert ist. • **`enabled`**— TTL-Überwachung ist aktiviert. • **`disabled`**— Die TTL-Überwachung ist deaktiviert.

Änderung der Amazon DocumentDB-Cluster-Parameter

In Amazon DocumentDB bestehen Cluster-Parametergruppen aus Parametern, die für alle Instances gelten, die Sie im Cluster erstellen. Im Fall benutzerdefinierter Cluster-Parametergruppen können Sie für alle von Ihnen erstellten Parametergruppen die Parameterwerte jederzeit ändern oder alle Parameterwerte jederzeit auf die Standardwerte für Parametergruppen zurücksetzen. In diesem Abschnitt wird beschrieben, wie Sie die Parameter, aus denen eine Amazon DocumentDB-Cluster-Parametergruppe besteht, und ihre Werte anzeigen und wie Sie diese Werte ändern oder aktualisieren können.

Parameter können dynamisch oder statisch sein. Wenn Sie einen dynamischen Parameter ändern und die Cluster-Parametergruppe speichern, wird die Änderung sofort übernommen, ungeachtet der

Einstellung für `Apply Immediately`. Wenn Sie einen statischen Parameter ändern und die Cluster-Parametergruppe speichern, wird die Änderung des Parameters erst nach einem manuellen Neustart der Instances angewendet.

Parameter einer Amazon DocumentDB-Cluster-Parametergruppe anzeigen

Sie können die Parameter eines Amazon DocumentDB-Clusters und ihre Werte mithilfe von oder sehen. AWS-Managementkonsole AWS CLI

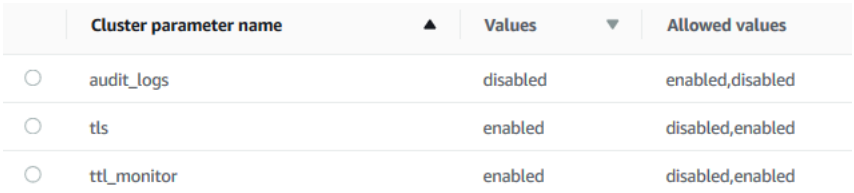
Using the AWS-Managementkonsole

So zeigen Sie die Details einer Cluster-Parametergruppe an

1. Melden Sie sich bei und öffnen Sie die Amazon DocumentDB-Konsole unter. AWS-Managementkonsole https://console.aws.amazon.com/docdb
2. Wählen Sie im Navigationsbereich Parameter groups (Parametergruppen) aus.

   > **Tip**
   >
   > Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Wählen Sie im Bereich Parameter groups (Parametergruppen) den Namen der Cluster-Parametergruppe aus, für die Sie die Details anzeigen möchten.
4. Auf der anschließend angezeigten Seite werden Ihnen für die einzelnen Parameter die folgenden Werte angezeigt: Name, aktueller Wert, zulässige Werte, ob der Parameter geändert werden kann, Anwendungstyp, Datentyp und Beschreibung.



Cluster parameter name	Values	Allowed values
audit_logs	disabled	enabled,disabled
tls	enabled	disabled,enabled
ttl_monitor	enabled	disabled,enabled

Using the AWS CLI

Um die Parameter einer Cluster-Parametergruppe und deren Werte anzuzeigen, verwenden Sie die `describe-db-cluster-parameters`-Operation mit den folgenden Parametern.

- **`--db-cluster-parameter-group-name`** – Erforderlich. Der Name der Cluster-Parametergruppe, für die Sie eine detaillierte Parameterliste möchten.
- **`--source`**— Fakultativ. Wenn angegeben, werden nur Parameter für eine bestimmte Quelle zurückgegeben. Parameterquellen können `engine-default`, `system` oder `user` sein.

Example

Der folgende Code listet die Parameter und deren Werte für die Parametergruppe `custom3-6-param-grp` auf. Um weitere Informationen zur Parametergruppe zu erhalten, lassen Sie die Zeile `--query` aus. Um Informationen zu allen Parametergruppen zu erhalten, lassen Sie die Zeile `--db-cluster-parameter-group-name` aus.

Für Linux, macOS oder Unix:

```
aws docdb describe-db-cluster-parameters \
   --db-cluster-parameter-group-name custom3-6-param-grp \
   --query 'Parameters[*].[ParameterName,ParameterValue]'
```

Für Windows:

```
aws docdb describe-db-cluster-parameters ^
   --db-cluster-parameter-group-name custom3-6-param-grp ^
   --query 'Parameters[*].[ParameterName,ParameterValue]'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
[
    [
        "audit_logs",
        "disabled"
    ],
    [
        "tls",
        "enabled"
    ],
    [
        "ttl_monitor",
        "enabled"
    ]
]
```

Ändern der Parameter einer Amazon DocumentDB-Cluster-Parametergruppe

Sie können die Parameter einer Parametergruppe mithilfe von AWS-Managementkonsole oder AWS CLIändern.

Using the AWS-Managementkonsole

So aktualisieren Sie die Parameter einer Cluster-Parametergruppe

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Wählen Sie im Navigationsbereich Parameter groups (Parametergruppen) aus.

   > **Tip**
   >
   > Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Wählen Sie im Bereich Parameter groups (Parametergruppen) die Cluster-Parametergruppe aus, deren Parameter Sie aktualisieren möchten.
4. Auf der resultierenden Seite werden die Parameter und die entsprechenden Details zu dieser Cluster-Parametergruppe angezeigt. Wählen Sie einen zu aktualisierenden Parameter aus.
5. Klicken Sie oben rechts auf der Seite auf Edit (Bearbeiten), um den Wert des Parameters zu ändern. Weitere Informationen zu den Typen von Cluster-Parametern finden Sie unter Referenz zu den Amazon DocumentDB-Clusterparametern.
6. Führen Sie die Änderung aus. Wählen Sie anschließend Modify cluster parameter (Cluster-Parameter ändern) aus, um die Änderungen zu speichern. Um die Änderungen zu verwerfen, wählen Sie Cancel (Abbrechen) aus.

Using the AWS CLI

Um die Parameter einer Cluster-Parametergruppe zu ändern, verwenden Sie die `modify-db-cluster-parameter-group`-Operation mit den folgenden Parametern:

- **`--db-cluster-parameter-group-name`** – Erforderlich. Der Name der Cluster-Parametergruppe, die Sie ändern.

- **`--parameters`** – Erforderlich. Der Parameter oder die Parameter, die Sie ändern. Jeder Parametereintrag muss Folgendes enthalten:
  - **`ParameterName`**— Der Name des Parameters, den Sie ändern.
  - **`ParameterValue`**— Der neue Wert für diesen Parameter.
  - **`ApplyMethod`**— Wie Sie Änderungen an diesem Parameter anwenden möchten. Zugelassene Werte sind `immediate` und `pending-reboot`.

    > **Note**
    >
    > Parameter mit dem `ApplyType` von `static` müssen über einen `ApplyMethod` von `pending-reboot` verfügen.

Um die Werte der Parameter einer Cluster-Parametergruppe zu ändern (AWS CLI)

Im folgenden Beispiel wird der Parameter `tls` geändert.

1. Listen Sie die Parameter von **`sample-parameter-group`** und ihre Werte auf.

   Für Linux, macOS oder Unix:

   ```
   aws docdb describe-db-cluster-parameters \
       --db-cluster-parameter-group-name sample-parameter-group
   ```

   Für Windows:

   ```
   aws docdb describe-db-cluster-parameters ^
       --db-cluster-parameter-group-name sample-parameter-group
   ```

   Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

   ```
   {
       "Parameters": [
           {
               "Source": "system",
               "ApplyType": "static",
               "AllowedValues": "disabled,enabled",
               "ParameterValue": "enabled",
               "ApplyMethod": "pending-reboot",
   ```

```
            "DataType": "string",
            "ParameterName": "tls",
            "IsModifiable": true,
            "Description": "Config to enable/disable TLS"
        },
        {
            "Source": "user",
            "ApplyType": "dynamic",
            "AllowedValues": "disabled,enabled",
            "ParameterValue": "enabled",
            "ApplyMethod": "pending-reboot",
            "DataType": "string",
            "ParameterName": "ttl_monitor",
            "IsModifiable": true,
            "Description": "Enables TTL Monitoring"
        }
    ]
}
```

2. Ändern Sie den **`tls`**-Parameter, sodass sein Wert **`disabled`** ist. Sie können den `ApplyMethod` nicht ändern, da der `ApplyType` `static` ist.

   Für Linux, macOS oder Unix:

```
aws docdb modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name sample-parameter-group \
    --parameters
 "ParameterName"=tls,ParameterValue=disabled,ApplyMethod=pending-reboot"
```

   Für Windows:

```
aws docdb modify-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name sample-parameter-group ^
    --parameters "ParameterName=tls,ParameterValue=disabled,ApplyMethod=pending-
reboot"
```

   Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    "DBClusterParameterGroupName": "sample-parameter-group"
}
```

3. Warten Sie mindestens 5 Minuten.
4. Listen Sie die Parameterwerte von **`sample-parameter-group`** auf.

   Für Linux, macOS oder Unix:

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

   Für Windows:

```
aws docdb describe-db-cluster-parameters ^
    --db-cluster-parameter-group-name sample-parameter-group
```

   Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    "Parameters": [
        {
            "ParameterName": "audit_logs",
            "ParameterValue": "disabled",
            "Description": "Enables auditing on cluster.",
            "Source": "system",
            "ApplyType": "dynamic",
            "DataType": "string",
            "AllowedValues": "enabled,disabled",
            "IsModifiable": true,
            "ApplyMethod": "pending-reboot"
        },
        {
            "ParameterName": "tls",
            "ParameterValue": "disabled",
            "Description": "Config to enable/disable TLS",
            "Source": "user",
            "ApplyType": "static",
            "DataType": "string",
            "AllowedValues": "disabled,enabled",
            "IsModifiable": true,
            "ApplyMethod": "pending-reboot"
        }
    ]
}
```

Grundlegendes zu Amazon DocumentDB DocumentDB-Endpunkten

Sie können Amazon DocumentDB DocumentDB-Endpunkte (mit MongoDB-Kompatibilität) verwenden, um eine Verbindung zu einem Cluster oder einer Instance herzustellen. Amazon DocumentDB hat drei verschiedene Arten von Endpunkten, von denen jeder seinen eigenen Zweck hat.

Themen

- Die Endpunkte eines Clusters finden
- Den Endpunkt einer Instanz finden
- Verbindung zu Endpunkten herstellen

Cluster-Endpunkt

Ein Cluster-Endpunkt ist ein Endpunkt für einen Amazon DocumentDB-Cluster, der eine Verbindung zur aktuellen primären Instance für den Cluster herstellt. Jeder Amazon DocumentDB-Cluster hat einen einzelnen Cluster-Endpunkt und eine primäre Instance. Im Falle eines Failovers wird der Cluster-Endpunkt auf die neue primäre Instance umgeschaltet.

Leser-Endpunkt

Ein Reader-Endpunkt ist ein Endpunkt für einen Amazon DocumentDB-Cluster, der eine Verbindung zu einem der verfügbaren Replikate für diesen Cluster herstellt. Jeder Amazon DocumentDB-Cluster hat einen Leser-Endpunkt. Wenn es mehr als ein Replikat gibt, leitet der Leser-Endpunkt jede Verbindungsanfrage an eines der Amazon DocumentDB DocumentDB-Replikate weiter.

Instance-Endpunkt

Ein Instance-Endpunkt ist ein Endpunkt, der sich mit einer bestimmten Instance verbindet. Jede Instance in einem Cluster, unabhängig davon, ob es sich um eine primäre oder eine Replikat-Instance handelt, hat ihren eigenen eindeutigen Instance-Endpunkt. Sie sollten keine Instance-Endpunkte in Ihrer Anwendung verwenden. Denn diese können im Falle eines Failovers Rollen ändern, sodass Code-Änderungen in Ihrer Anwendung erforderlich sind.

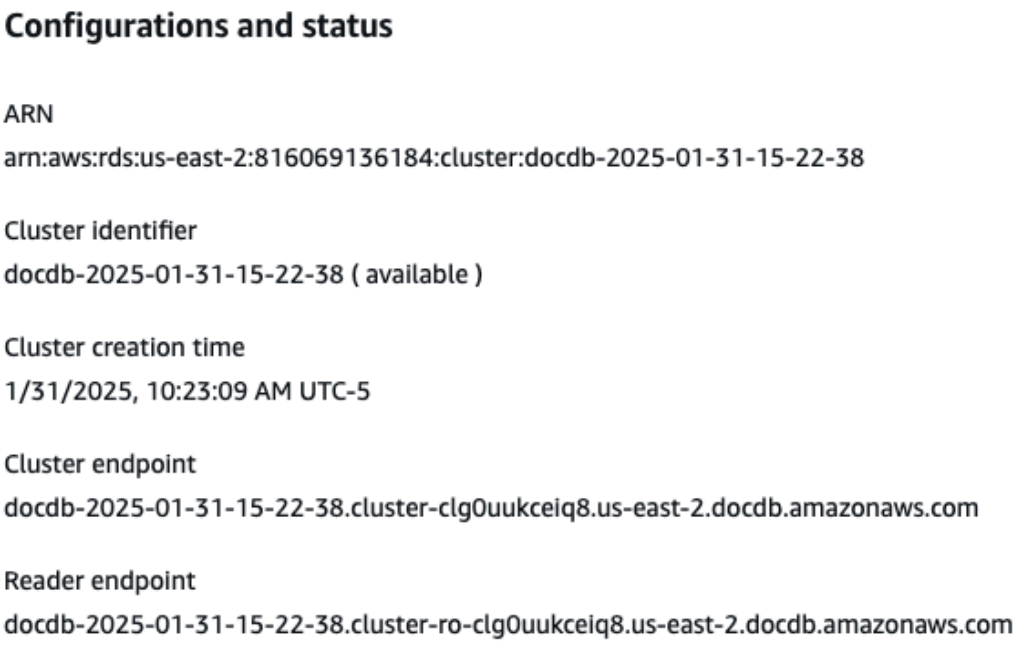
Die Endpunkte eines Clusters finden

Sie können den Cluster-Endpunkt und den Reader-Endpunkt eines Clusters mithilfe der Amazon DocumentDB DocumentDB-Konsole oder AWS CLI finden.

Using the AWS-Managementkonsole

Um die Endpunkte eines Clusters mithilfe der Konsole zu finden:

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Klicken Sie im Navigationsbereich auf Cluster.
3. Wählen Sie aus der Liste der Cluster den Namen des Clusters, der Sie interessiert.
4. Wählen Sie auf der Seite mit den Cluster-Details die Registerkarte Konfiguration aus. Im Abschnitt Konfigurationen und Status finden Sie den Cluster-Endpunkt und den Reader-Endpunkt.



5. Um eine Verbindung zu diesem Cluster herzustellen, wählen Sie die Registerkarte Konnektivität und Sicherheit. Suchen Sie die Verbindungszeichenfolge für die `mongo` Shell und die Verbindungszeichenfolge, die im Anwendungscode verwendet werden kann, um eine Verbindung zu Ihrem Cluster herzustellen.



Using the AWS CLI

Um die Cluster- und Reader-Endpunkte für Ihren Cluster mithilfe von zu finden AWS CLI, führen Sie den `describe-db-clusters` Befehl mit diesen Parametern aus.

Parameters

- **`--db-cluster-identifier`**— Optional. Gibt den Cluster an, für den die Endpunkte zurückgegeben werden sollen. Wenn diese Option weggelassen wird, werden Endpunkte für bis zu 100 Ihrer Cluster zurückgegeben.
- **`--query`**—Fakultativ. Gibt die anzuzeigenden Felder an. Dies ist hilfreich, da die Datenmenge, die Sie anzeigen müssen, um die Endpunkte zu finden, reduziert wird. Wenn diese Option weggelassen wird, werden alle Informationen über einen Cluster zurückgegeben.
- **`--region`**—Fakultativ. Verwenden Sie den Parameter `--region`, um die Region anzugeben, für die Sie den Befehl anwenden möchten. Wenn diese Angabe weggelassen wird, wird Ihre Standardregion verwendet.

Example

Das folgende Beispiel liefert die Werte `DBClusterIdentifier`, Endpunkt (Cluster-Endpunkt) und `ReaderEndpoint` für `sample-cluster`.

Für Linux, macOS oder Unix:

```
aws docdb describe-db-clusters \
   --region us-east-1 \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[DBClusterIdentifier,Port,Endpoint,ReaderEndpoint]'
```

Für Windows:

```
aws docdb describe-db-clusters ^
   --region us-east-1 ^
   --db-cluster-identifier sample-cluster ^
   --query 'DBClusters[*].[DBClusterIdentifier,Port,Endpoint,ReaderEndpoint]'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
[
  [
     "sample-cluster",
     27017,
     "sample-cluster.cluster-corlsfccjozr.us-east-1.docdb.amazonaws.com",
     "sample-cluster.cluster-ro-corlsfccjozr.us-east-1.docdb.amazonaws.com"
  ]
]
```

Nachdem Sie nun den Cluster-Endpunkt haben, können Sie sich mit `mongo` oder `mongodb` mit dem Cluster verbinden. Weitere Informationen finden Sie unter Verbindung zu Endpunkten herstellen.

Den Endpunkt einer Instanz finden

Sie können den Endpunkt für eine Instance mithilfe der Amazon DocumentDB DocumentDB-Konsole oder der AWS CLI finden.

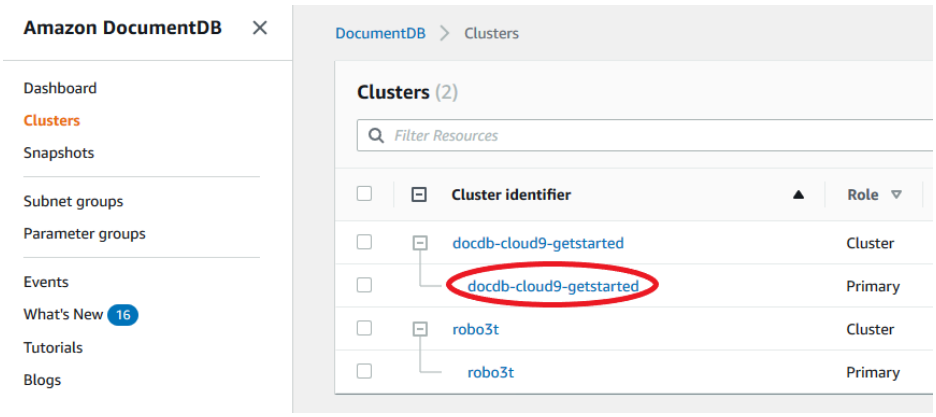
Using the AWS-Managementkonsole

So suchen Sie den Endpunkt einer Instance über die Konsole:

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Klicken Sie im Navigationsbereich auf Cluster.

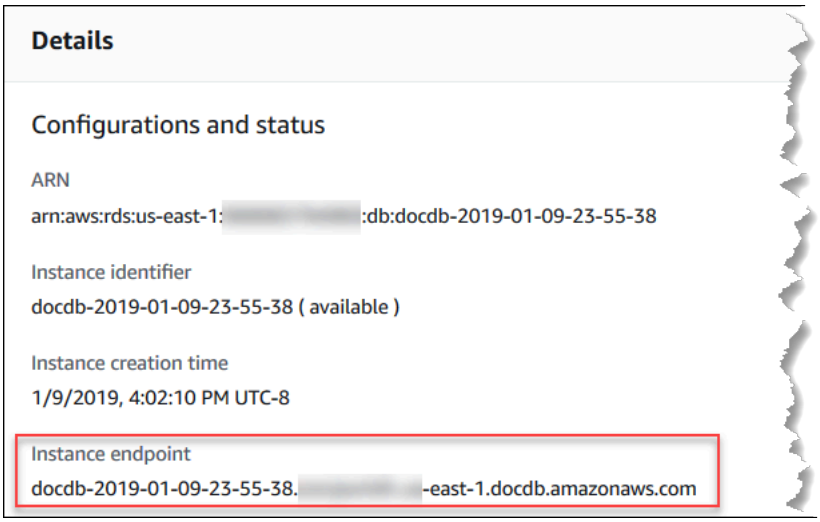
> **Tip**
>
> Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Im Cluster-Navigationsfeld sehen Sie die Spalte Cluster-ID. Ihre Instances werden unter Clustern aufgeführt, ähnlich wie im folgenden Screenshot.



4. Markieren Sie das Kästchen links neben der Instanz, an der Sie interessiert sind.

5. Scrollen Sie nach unten zum Abschnitt Details und suchen Sie dann den Instance-Endpunkt.



6. Um eine Verbindung zu dieser Instance herzustellen, scrollen Sie nach oben zum Abschnitt Connect (Verbinden). Suchen Sie die Verbindungszeichenfolge für die mongo-Shell und eine Verbindungszeichenfolge, die im Anwendungscode verwendet werden kann, um eine Verbindung zu Ihrer Instance herzustellen.



Using the AWS CLI

Führen Sie den folgenden Befehl mit diesen Argumenten aus AWS CLI, um den Instanzendpunkt mithilfe von zu finden.

Argumente

- **`--db-instance-identifier`**— Optional. Gibt die Instance an, für die der Endpunkt zurückgegeben werden soll. Wenn diese Option weggelassen wird, wird der Endpunkt für bis zu 100 Ihrer Instances zurückgegeben.
- **`--query`**—Fakultativ. Gibt die anzuzeigenden Felder an. Dies ist hilfreich, da die Datenmenge, die Sie anzeigen müssen, um die Endpunkte zu finden, reduziert wird. Wenn diese Option weggelassen wird, werden alle Informationen zu einer Instance zurückgegeben. Das Feld `Endpoint` hat drei Mitglieder, sodass die Auflistung in der Abfrage wie im folgenden Beispiel alle drei Mitglieder zurückgibt. Wenn Sie nur an einigen der `Endpoint`-Mitglieder interessiert sind, ersetzen Sie `Endpoint` in der Abfrage durch die Mitglieder, die Sie interessieren, wie im zweiten Beispiel.
- **`--region`**—Fakultativ. Verwenden Sie den Parameter `--region`, um die Region anzugeben, für die Sie den Befehl anwenden möchten. Wenn diese Angabe weggelassen wird, wird Ihre Standardregion verwendet.

Example

Für Linux, macOS oder Unix:

```
aws docdb describe-db-instances \
    --region us-east-1 \
    --db-instance-identifier sample-cluster-instance \
    --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint]'
```

Für Windows:

```
aws docdb describe-db-instances ^
    --region us-east-1 ^
    --db-instance-identifier sample-cluster-instance ^
    --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint]'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
[
    [
        "sample-cluster-instance",
        {
            "Port": 27017,
            "Address": "sample-cluster-instance.corcjozrlsfc.us-
east-1.docdb.amazonaws.com",
            "HostedZoneId": "Z2R2ITUGPM61AM"
        }
    ]
]
```

Wenn Sie die Ausgabe reduzieren, um die `HostedZoneId` des Endpunkts zu eliminieren, können Sie Ihre Abfrage ändern, indem Sie `Endpoint.Port` und `Endpoint.Address` festlegen.

Für Linux, macOS oder Unix:

```
aws docdb describe-db-instances \
    --region us-east-1 \
    --db-instance-identifier sample-cluster-instance \
    --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint.Port,Endpoint.Address]'
```

Für Windows:

```
aws docdb describe-db-instances ^
    --region us-east-1 ^
    --db-instance-identifier sample-cluster-instance ^
    --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint.Port,Endpoint.Address]'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
[
    [
        "sample-cluster-instance",
        27017,
        "sample-cluster-instance.corcjozrlsfc.us-east-1.docdb.amazonaws.com"
    ]
]
```

Nachdem Sie nun den Instance-Endpunkt haben, können Sie sich mit mongo oder mongodb mit der Instance verbinden. Weitere Informationen finden Sie unter Verbindung zu Endpunkten herstellen.

Verbindung zu Endpunkten herstellen

Wenn Sie Ihren Endpunkt, entweder Cluster oder Instance, haben, können Sie sich über die mongo-Shell oder eine Verbindungszeichenfolge mit ihm verbinden.

Verbindung über die Mongo-Shell herstellen

Verwenden Sie die folgende Struktur, um die Zeichenfolge zu erstellen, die Sie für die Verbindung zu Ihrem Cluster oder Ihrer Instance über die mongo-Shell benötigen:

```
mongo \
    --ssl \
    --host Endpoint:Port \
    --sslCAFile global-bundle.pem \
    --username UserName \
    --password Password
```

Beispiele für **Mongo-Shell**

Verbinden mit einem Cluster:

```
mongo \
    --ssl \
    --host sample-cluster.corcjozrlsfc.us-east-1.docdb.amazonaws.com:27017 \
    --sslCAFile global-bundle.pem \
    --username UserName \
    --password Password
```

Verbindung zu einer Instance herstellen:

```
mongo \
    --ssl \
    --host sample-cluster-instance.corcjozrlsfc.us-east-1.docdb.amazonaws.com:27017 \
    --sslCAFile global-bundle.pem \
    --username UserName \
```

```
    --password Password
```

Verbindung über eine Verbindungszeichenfolge herstellen

Verwenden Sie die folgende Struktur, um die Verbindungszeichenfolge zu erstellen, die Sie für die Verbindung zu Ihrem Cluster oder Ihrer Instance benötigen.

```
mongodb://UserName:Password@endpoint:port?replicaSet=rs0&ssl_ca_certs=global-
bundle.pem
```

Beispiele für Verbindungszeichenfolgen

Verbinden mit einem Cluster:

```
mongodb://UserName:Password@sample-cluster.cluster-corlsfccjozr.us-
east-1.docdb.amazonaws.com:27017?replicaSet=rs0&ssl_ca_certs=global-bundle.pem
```

Verbindung zu einer Instance herstellen:

```
mongodb://UserName:Password@sample-cluster-instance.cluster-corlsfccjozr.us-
east-1.docdb.amazonaws.com:27017?replicaSet=rs0&ssl_ca_certs=global-bundle.pem
```

Grundlegendes zu Amazon DocumentDB Amazon Resource Names (ARNs)

Ressourcen, die Sie in erstellen, AWS werden jeweils eindeutig mit einem Amazon-Ressourcennamen (ARN) identifiziert. Für bestimmte Amazon DocumentDB DocumentDB-Operationen (mit MongoDB-Kompatibilität) müssen Sie eine Amazon DocumentDB DocumentDB-Ressource eindeutig identifizieren, indem Sie ihren ARN angeben. Wenn Sie beispielsweise ein Tag zu einer Ressource hinzufügen, müssen Sie den ARN der Ressource angeben.

Themen

- Einen ARN für eine Amazon DocumentDB DocumentDB-Ressource erstellen
- Einen ARN für eine Amazon DocumentDB DocumentDB-Ressource finden

Einen ARN für eine Amazon DocumentDB DocumentDB-Ressource erstellen

Sie können einen ARN für eine Amazon DocumentDB DocumentDB-Ressource mit der folgenden Syntax erstellen. Amazon DocumentDB verwendet das Format von Amazon Relational Database Service (Amazon RDS) ARNs. Amazon DocumentDB DocumentDB-ARNs enthalten `rds` und nicht. `docdb`

`arn:aws:rds:`*`region`*`:`*`account_number`*`:`*`resource_type`*`:`*`resource_id`*

Name der Region	Region	Verfügbarkeitszonen (Datenverarbeitung)
USA Ost (Ohio)	`us-east-2`	3
USA Ost (Nord-Virginia)	`us-east-1`	6
USA West (Oregon)	`us-west-2`	4
Afrika (Kapstadt)	`af-south-1`	3
Südamerika (São Paulo)	`sa-east-1`	3
Asien-Pazifik (Hongkong)	`ap-east-1`	3
Asien-Pazifik (Hyderabad)	`ap-south-2`	3
Asien-Pazifik (Malaysia)	`ap-southeast-5`	3
Asien-Pazifik (Mumbai)	`ap-south-1`	3
Asien-Pazifik (Osaka)	`ap-northeast-3`	3
Asien-Pazifik (Seoul)	`ap-northeast-2`	4

Name der Region	Region	Verfügbarkeitszonen (Datenverarbeitung)
Asien-Pazifik (Singapur)	ap-southeast-1	3
Asien-Pazifik (Sydney)	ap-southeast-2	3
Asien-Pazifik (Jakarta)	ap-southeast-3	3
Asien-Pazifik (Melbourne)	ap-southeast-4	3
Asien-Pazifik (Thailand)	ap-southeast-7	3
Asien-Pazifik (Tokio)	ap-northeast-1	3
Kanada (Zentral)	ca-central-1	3
Kanada West (Calgary)	ca-west-1	3
Region China (Peking)	cn-north-1	3
China (Ningxia)	cn-northwest-1	3
Europa (Frankfurt)	eu-central-1	3
Europa (Zürich)	eu-central-2	3
Europa (Irland)	eu-west-1	3
Europa (London)	eu-west-2	3
Europa (Milan)	eu-south-1	3
Europa (Paris)	eu-west-3	3

Name der Region	Region	Verfügbarkeitszonen (Datenverarbeitung)
Europa (Spain)	`eu-south-2`	3
Europa (Stockholm)	`eu-north-1`	3
Mexiko (Zentral)	`mx-central-1`	3
Naher Osten (VAE)	`me-central-1`	3
Israel (Tel Aviv)	`il-central-1`	3
AWS GovCloud (US-West)	`us-gov-west-1`	3
AWS GovCloud (US-East)	`us-gov-east-1`	3

> **Note**
>
> Die Amazon DocumentDB DocumentDB-Architektur trennt Speicher und Datenverarbeitung. Für die Speicherebene repliziert Amazon DocumentDB sechs Kopien Ihrer Daten in drei AWS Availability Zones (AZs). Die in der obigen Tabelle aufgeführten AZs geben die Anzahl der AZs an, die Sie in einer bestimmten Region zur Bereitstellung von Compute-Instances verwenden können. Wenn Sie beispielsweise einen Amazon DocumentDB-Cluster in ap-northeast-1 starten, wird Ihr Speicher auf sechs Arten über drei AZs repliziert, Ihre Compute-Instances sind jedoch nur in zwei AZs verfügbar.

Die folgende Tabelle zeigt das Format, das Sie verwenden sollten, wenn Sie einen ARN für eine bestimmte Amazon DocumentDB DocumentDB-Ressource erstellen. Amazon DocumentDB teilt das Format von Amazon RDS-ARNs. Amazon DocumentDB DocumentDB-ARNs enthalten `rds` und nicht. `docdb`

Ressourcentyp	ARN-Format/Beispiel
Instance (db)	`arn:aws:rds:` *`region`*`:`*`account_number`*`:db:`*`resource_id`*

Ressourcentyp	ARN-Format/Beispiel
	`arn:aws:rds:us-east-1:` *1234567890* `:db:`*sample-db-instance*
Cluster (`cluster`)	`arn:aws:rds:` *region*`:`*account_number* `:cluster:` *resource_id* `arn:aws:rds:us-east-1:` *1234567890* `:cluster:` *sample-db-cluster*
Cluster-Parametergruppe (`cluster-pg`)	`arn:aws:rds:` *region*`:`*account_number* `:cluster-pg:` *resource_id* `arn:aws:rds:us-east-1:` *1234567890* `:cluster-pg:` *sample-db-cluster-parameter-group*
Sicherheitsgruppe (`secgrp`)	`arn:aws:rds:` *region*`:`*account_number* `:secgrp:`*resource_id* `arn:aws:rds:us-east-1:` *1234567890* `:secgrp:`*sample-public-secgrp*
Cluster-Snapshot (`cluster-snapshot`)	`arn:aws:rds:` *region*`:`*account_number* `:cluster-snapshot:` *resource_id* `arn:aws:rds:us-east-1:` *1234567890* `:cluster-snapshot:` *sample-db-cluster-snapshot*
Subnetzgruppe (`subgrp`)	`arn:aws:rds:` *region*`:`*account_number* `:subgrp:`*resource_id* `arn:aws:rds:us-east-1:` *1234567890* `:subgrp:`*sample-subnet-10*

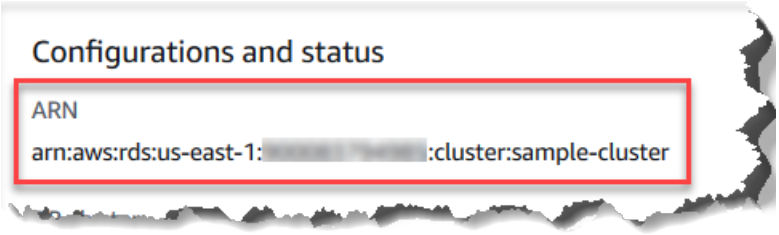
Einen ARN für eine Amazon DocumentDB DocumentDB-Ressource finden

Sie können den ARN einer Amazon DocumentDB DocumentDB-Ressource mit dem AWS-Managementkonsole oder dem AWS CLI ermitteln.

Using the AWS-Managementkonsole

Um einen ARN über die Konsole zu suchen, navigieren Sie zu der Ressource, für die Sie einen ARN erhalten möchten, und zeigen Sie die Details für diese Ressource an.

Sie können beispielsweise den ARN für einen Cluster abrufen, indem Sie auf der Cluster-Detailseite die Registerkarte Konfiguration auswählen. Die ARN finden Sie im Abschnitt Konfigurationen und Status, wie im folgenden Screenshot gezeigt.



Using the AWS CLI

Um einen ARN mit der AWS CLI für eine bestimmte Amazon DocumentDB DocumentDB-Ressource zu erhalten, verwenden Sie den `describe` Vorgang für diese Ressource. Die folgende Tabelle zeigt jeden AWS CLI Vorgang und die ARN-Eigenschaft, die mit dem Vorgang zum Abrufen eines ARN verwendet wird.

AWS CLI Befehl	ARN-Eigenschaft
`describe-db-instances`	`DBInstanceArn`
`describe-db-clusters`	`DBClusterArn`
`describe-db-parameter-groups`	`DBParameterGroupArn`
`describe-db-cluster-parameter-groups`	`DBClusterParameterGroupArn`
`describe-db-security-groups`	`DBSecurityGroupArn`
`describe-db-snapshots`	`DBSnapshotArn`

AWS CLI Befehl	ARN-Eigenschaft
`describe-db-cluster-snapshots`	`DBClusterSnapshotArn`
`describe-db-subnet-groups`	`DBSubnetGroupArn`

Example- ARN Ihres Clusters suchen

Der folgende AWS CLI Vorgang findet den ARN für den Cluster`sample-cluster`.

Für Linux, macOS oder Unix:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].DBClusterArn'
```

Für Windows:

```
aws docdb describe-db-clusters ^
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].DBClusterArn'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
[
    "arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster"
]
```

Example- ARNs für mehrere Parametergruppen suchen

Für Linux, macOS oder Unix:

```
aws docdb describe-db-cluster-parameter-groups \
   --query 'DBClusterParameterGroups[*].DBClusterParameterGroupArn'
```

Für Windows:

```
aws docdb describe-db-cluster-parameter-groups ^
```

```
    --query 'DBClusterParameterGroups[*].DBClusterParameterGroupArn'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
[
    "arn:aws:rds:us-east-1:123456789012:cluster-pg:custom3-6-param-grp",
    "arn:aws:rds:us-east-1:123456789012:cluster-pg:default.aurora5.6",
    "arn:aws:rds:us-east-1:123456789012:cluster-pg:default.docdb3.6"
]
```

Taggen von Amazon DocumentDB DocumentDB-Ressourcen

Sie können Amazon DocumentDB-Tags (mit MongoDB-Kompatibilität) verwenden, um Metadaten zu Ihren Amazon DocumentDB DocumentDB-Ressourcen hinzuzufügen. Diese Tags können zusammen mit AWS Identity and Access Management (IAM-) Richtlinien verwendet werden, um den Zugriff auf Amazon DocumentDB DocumentDB-Ressourcen zu verwalten und zu kontrollieren, welche Aktionen auf die Ressourcen angewendet werden können. Sie können Tags auch verwenden, um Kosten zu verfolgen, indem Ausgaben für ähnlich markierte Ressourcen gruppiert werden.

Sie können die folgenden Amazon DocumentDB DocumentDB-Ressourcen taggen:

- Cluster
- Instances
- -Snapshots
- Cluster-Snapshots
- Parametergruppen
- Cluster-Parametergruppen
- Sicherheitsgruppen
- Subnetzgruppen

Übersicht über Amazon DocumentDB DocumentDB-Ressourcen-Tags

Ein Amazon DocumentDB-Tag ist ein Name-Wert-Paar, das Sie definieren und mit einer Amazon DocumentDB DocumentDB-Ressource verknüpfen. Der Name wird als der Schlüssel bezeichnet. Die Angabe eines Wertes für den Schlüssel ist optional. Sie können Tags verwenden, um einer Amazon DocumentDB DocumentDB-Ressource beliebige Informationen zuzuweisen. Sie können

einen Tag-Schlüssel z. B. dazu verwenden, um eine Kategorie zu definieren, und ein Tag-Wert könnte ein Element in dieser Kategorie sein. Sie könnten beispielsweise einen Tag-Schlüssel `project` und einen Tag-Wert von definieren`Salix`, was angibt, dass die Amazon DocumentDB DocumentDB-Ressource dem Salix-Projekt zugewiesen ist. Sie können Tags auch verwenden, um Amazon DocumentDB DocumentDB-Ressourcen so zu kennzeichnen, dass sie für Test- oder Produktionszwecke verwendet werden, indem Sie einen Schlüssel wie `environment=test` oder verwenden. `environment=production` Wir empfehlen Ihnen, einen konsistenten Satz von Tag-Schlüsseln zu verwenden, um die Nachverfolgung von Metadaten zu erleichtern, die Amazon DocumentDB DocumentDB-Ressourcen zugeordnet sind.

Sie können Tags verwenden, um Ihre AWS Rechnung so zu organisieren, dass sie Ihrer eigenen Kostenstruktur entspricht. Melden Sie sich dazu an, um Ihre AWS-Konto Rechnung mit den Tag-Schlüsselwerten zu erhalten. Um dann die Kosten kombinierter Ressourcen anzuzeigen, organisieren Sie Ihre Fakturierungsinformationen nach Ressourcen mit gleichen Tag-Schlüsselwerten. Beispielsweise können Sie mehrere Ressourcen mit einem bestimmten Anwendungsnamen markieren und dann Ihre Fakturierungsinformationen so organisieren, dass Sie die Gesamtkosten dieser Anwendung über mehrere Services hinweg sehen können. Weitere Informationen finden Sie unter Verwendung von Kostenzuordnungs-Tags im AWS Benutzerhandbuch für Fakturierung und Kostenmanagement.

Jede Amazon DocumentDB DocumentDB-Ressource hat einen Tagsatz, der alle Tags enthält, die dieser Ressource zugewiesen sind. Ein Tag-Satz kann bis zu zehn Tags enthalten oder leer sein. Wenn Sie einer Amazon DocumentDB DocumentDB-Ressource ein Tag hinzufügen, das denselben Schlüssel wie ein vorhandenes Tag auf einer Ressource hat, überschreibt der neue Wert den alten Wert.

AWS weist Ihren Tags keine semantische Bedeutung zu; Tags werden ausschließlich als Zeichenketten interpretiert. Amazon DocumentDB kann Tags für eine Instance oder andere Amazon DocumentDB DocumentDB-Ressourcen festlegen, abhängig von den Einstellungen, die Sie bei der Erstellung der Ressource verwenden. Amazon DocumentDB könnte beispielsweise ein Tag hinzufügen, das angibt, dass eine Instance für die Produktion oder zum Testen bestimmt ist.

Sie können ein Tag an einen Snapshot anfügen. Diese Gruppierung taucht jedoch nicht in Ihrer Rechnung auf.

Sie können das AWS-Managementkonsole oder das verwenden, um Tags AWS CLI zu Amazon DocumentDB DocumentDB-Ressourcen hinzuzufügen, aufzulisten und zu löschen. Wenn Sie den verwenden AWS CLI, müssen Sie den Amazon-Ressourcennamen (ARN) für die Ressource angeben, mit der Sie arbeiten möchten. Weitere Informationen zu Amazon DocumentDB

DocumentDB-ARNs finden Sie unter. Grundlegendes zu Amazon DocumentDB Amazon Resource Names (ARNs)

Tag-Einschränkungen

Die folgenden Einschränkungen gelten für Amazon DocumentDB-Tags:

- Maximale Anzahl von Tags pro Ressource: 10
- Maximale Schlüssellänge – 128 Unicode-Zeichen
- Maximale Wertlänge – 256 Unicode-Zeichen
- Gültige Zeichen für Schlüssel und Wert — Groß- und Kleinbuchstaben im UTF-8 Zeichensatz, Ziffern, Leerzeichen und die folgenden Zeichen: `_ . : / = + -` und `@` (Java-Regex:) `"^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$"`
- Bei Tag-Schlüsseln und -Werten muss die Groß- und Kleinschreibung beachtet werden.
- Das Präfix `aws:` kann nicht für die Tag-Schlüssel oder Werte verwendet werden. Dieses Präfix ist für AWS reserviert.

Hinzufügen und Aktualisieren von Tags in einer Amazon DocumentDB DocumentDB-Ressource

Mit dem AWS-Managementkonsole oder dem können Sie einer Ressource bis zu 10 Tags hinzufügen. AWS CLI

Using the AWS-Managementkonsole

Der Prozess für das Hinzufügen eines Tags zu einer Ressource ist ähnlich, unabhängig davon, welcher Ressource Sie das Tag hinzufügen. In diesem Beispiel fügen Sie ein Tag zu einem Cluster hinzu.

So fügen Sie Tags einem Cluster über die Konsole hinzu oder aktualisieren sie

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Wählen Sie im Navigationsbereich Cluster aus.
3. Wählen Sie den Namen des Clusters aus, zu dem Sie Tags hinzufügen möchten.
4. Scrollen Sie nach unten zum Abschnitt Tags und wählen Sie anschließend Bearbeiten aus.

5. Für alle Tags, die Sie dieser Ressource hinzufügen möchten, gehen Sie wie folgt vor:

   a. Um ein neues Tag hinzuzufügen, geben Sie den Namen des Tags in das Feld Schlüssel ein. Um den Wert eines Tags zu ändern, suchen Sie den Namen des Tags in der Spalte Schlüssel.

   b. Um dem Tag einen neuen oder aktualisierten Wert zu geben, geben Sie einen Wert für das Tag in das Feld Wert ein.

   c. Wenn Sie mehrere Tags hinzufügen möchten, klicken Sie auf Hinzufügen. Andernfalls klicken Sie auf Speichern, sobald Sie fertig sind.

Using the AWS CLI

Der Prozess für das Hinzufügen eines Tags zu einer Ressource ist ähnlich, unabhängig davon, welcher Ressource Sie die Tags hinzufügen. In diesem Beispiel fügen Sie drei Tags zu einem Cluster hinzu. Das zweite Tag, `key2`, hat keinen Wert.

Verwenden Sie den AWS CLI Vorgang `add-tags-to-resource` mit diesen Parametern.

Parameters

- **`--resource-name`**— Der ARN der Amazon DocumentDB DocumentDB-Ressource, der Sie Tags hinzufügen möchten.
- **`--tags`**—Eine Liste der Tags (Schlüssel-Wert-Paar), die Sie dieser Ressource hinzufügen möchten, im folgenden Format. `Key=`*`key-name`*`,Value=`*`tag-value`*

Example

Für Linux, macOS oder Unix:

```
aws docdb add-tags-to-resource \
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster \
    --tags Key=key1,Value=value1 Key=key2 Key=key3,Value=value3
```

Für Windows:

```
aws docdb add-tags-to-resource ^
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster \
    --tags Key=key1,Value=value1 Key=key2 Key=key3,Value=value3
```

Die `add-tags-to-resource`-Operation erzeugt keine Ausgabe. Um die Ergebnisse der Operation anzuzeigen, verwenden Sie die `list-tags-for-resource`-Operation.

Tags in einer Amazon DocumentDB DocumentDB-Ressource auflisten

Sie können das AWS-Managementkonsole oder das verwenden AWS CLI , um eine Liste der Tags für eine Amazon DocumentDB DocumentDB-Ressource zu erhalten.

Using the AWS-Managementkonsole

Der Prozess für das Auflisten von Tags in einer Ressource ist ähnlich, unabhängig davon, welcher Ressource Sie das Tag hinzufügen. In diesem Beispiel listen Sie die Tags für einen Cluster auf.

So listen Sie die Tags in einem Cluster mithilfe der Konsole auf

1. Öffnen Sie die Amazon DocumentDB DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Wählen Sie im Navigationsbereich Cluster aus.
3. Wählen Sie den Namen des Clusters aus, für den Sie Tags auflisten möchten.
4. Um eine Liste der Tags in dieser Ressource anzuzeigen, scrollen Sie nach unten zum Abschnitt Tags.

Using the AWS CLI

Der Prozess für das Auflisten der Tags in einer Ressource ist ähnlich, unabhängig davon, für welche Ressource Sie das Tag auflisten. In diesem Beispiel listen Sie die Tags in einem Cluster auf.

Verwenden Sie die AWS CLI Operation `list-tags-for-resource` mit diesen Parametern.

Parameters

- **`--resource-name`**—Erforderlich. Der ARN der Amazon DocumentDB DocumentDB-Ressource, für die Sie Tags auflisten möchten.

Example

Für Linux, macOS oder Unix:

```
aws docdb list-tags-for-resource \
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster
```

Für Windows:

```
aws docdb list-tags-for-resource ^
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    "TagList": [
        {
            "Key": "key1",
            "Value": "value1"
        },
        {
            "Key": "key2",
            "Value": ""
        },
        {
            "Key": "key3",
            "Value": "value3"
        }
    ]
}
```

Tags aus einer Amazon DocumentDB DocumentDB-Ressource entfernen

Sie können das AWS-Managementkonsole oder das verwenden, AWS CLI um Tags aus Amazon DocumentDB DocumentDB-Ressourcen zu entfernen.

Using the AWS-Managementkonsole

Der Prozess für das Entfernen von Tags aus einer Ressource ist ähnlich, unabhängig davon, welcher Ressource Sie das Tag hinzufügen. In diesem Beispiel entfernen Sie Tags aus einem Cluster.

So entfernen Sie Tags aus einem Cluster mithilfe der Konsole

1. Öffnen Sie die Amazon DocumentDB DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Wählen Sie im Navigationsbereich Cluster aus.
3. Wählen Sie den Namen des Clusters aus, aus dem Sie Tags entfernen möchten.
4. Scrollen Sie nach unten zum Abschnitt Tags und wählen Sie anschließend Bearbeiten aus.
5. Wenn Sie alle Tags aus dieser Ressource entfernen möchten, wählen Sie Remove all (Alle entfernen) aus. Andernfalls gehen Sie für alle Tags, die Sie aus dieser Ressource entfernen möchten, wie folgt vor:

   a. Suchen Sie den Namen des Tags in der Spalte Schlüssel.
   b. Wählen Sie Entfernen auf derselben Zeile wie der Tag-Schlüssel aus.
   c. Wenn Sie fertig sind, wählen Sie Speichern aus.

Using the AWS CLI

Der Prozess für das Entfernen eines Tags aus einer Ressource ist ähnlich, unabhängig davon, aus welcher Ressource Sie das Tag entfernen. In diesem Beispiel entfernen Sie ein Tag aus einem Cluster.

Verwenden Sie die AWS CLI Operation `remove-tags-from-resource` mit diesen Parametern.

- **`--resource-name`**—Erforderlich. Der ARN der Amazon DocumentDB DocumentDB-Ressource, aus der Sie Tags entfernen möchten.
- **`--tag-keys`**—Erforderlich. Eine Liste der Tag-Schlüssel, die von dieser Ressource entfernt werden sollen.

Example

Für Linux, macOS oder Unix:

```
aws docdb remove-tags-from-resource \
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster \
    --tag-keys key1 key3
```

Für Windows:

```
aws docdb remove-tags-from-resource ^
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster \
    --tag-keys key1 key3
```

Die `removed-tags-from-resource`-Operation erzeugt keine Ausgabe. Um die Ergebnisse der Operation anzuzeigen, verwenden Sie die `list-tags-for-resource`-Operation.

Wartung von Amazon DocumentDB

Amazon DocumentDB führt regelmäßig zwei Arten von Wartungsarbeiten durch:

- Die Cluster-Wartung aktualisiert die Datenbank-Engine. Engine-Updates enthalten Sicherheitsupdates, Bugfixes, neue Funktionen und andere Engine-Verbesserungen.
- Bei der Instance-Wartung wird das Betriebssystem (OS) auf der Instance aktualisiert.

Engine-Patches und Betriebssystem-Updates verwenden dieselben drei Lebenszykluskategorien — optional , erforderlich und erzwungen — mit derselben Benachrichtigung und demselben Anwendungsverhalten für jede Kategorie. Engine-Versionen haben auch eine vierte Kategorie: Nebenversionen, auf die Sie manuell aktualisieren. Die Kategorien sind:

- Optional — enthält unkritische Verbesserungen. Kein automatisches Bewerbungsdatum und keine AHD-Benachrichtigung; bewerben Sie sich, wann es Ihnen passt. (Für Betriebssystem-Updates können Sie abonnieren, um benachrichtigt `RDS-EVENT-0230` zu werden, sobald eines verfügbar ist.)
- Erforderlich — enthält Sicherheitsupdates und andere wichtige Updates. Sie erhalten eine Benachrichtigung per Health Dashboard (AHD) und per E-Mail. Eine erforderliche Aktion wird automatisch während des Wartungsfensters Ihres Clusters oder Ihrer Instance angewendet, nachdem sie abgelaufen ist`AutoAppliedAfterDate`. Sie können einen Aufschub vornehmen, indem Sie das Wartungsfenster vor diesem Datum ändern.
- Erzwungen — eine seltene, äußerst kritische Lösung. Auto-applies außerhalb Ihres Wartungsfensters, nachdem es abgelaufen ist`ForcedApplyDate`. Amazon DocumentDB bezeichnet nur eine Aktion, die erzwungen wird, wenn keine andere Option verfügbar ist.
- Nebenversion (nur Engine-Versionen) — eine nummerierte Engine-Version, die einer Hauptversion hinzugefügt wird (z. B.). `5.0.1` User-driven: Sie führen ein Upgrade durch, indem Sie die Engine-Version des Clusters ändern. Gilt nie automatisch; keine AHD-Benachrichtigung. Für Hauptversionen vor 5.0 werden keine Nebenversionen veröffentlicht.

Engine-Patches werden in einer einzigen Kategorie (optional, erforderlich oder erzwungen) veröffentlicht und bleiben dort. Fortschritte bei Betriebssystemupdates: Die meisten sind zunächst optional und werden, wenn sie nicht angewendet werden, in die Kategorie „Erforderlich“ übergehen und schließlich in erzwungene Updates übergehen. Der genaue Zeitpunkt hängt vom Patch ab und wird in der AHD-Benachrichtigung und in den von zurückgegebenen Datumsfeldern veröffentlicht `describe-pending-maintenance-actions` (siehe Termine anwenden). In den Amazon DocumentDB-Versionshinweisen werden diese Kategorienamen verwendet, wenn Änderungen an der Engine angekündigt werden.

Durch das Anwenden eines Engine-Patches wird der Cluster kurzzeitig offline geschaltet. Im Rest dieses Themas wird beschrieben, wie Wartungsfenster funktionieren, wie Sie ausstehende Arbeiten finden, wie Engine-Patches und Nebenversionen angewendet werden, wie Betriebssystemupdates funktionieren und wie spezielle Verfahren für globale Cluster behandelt werden.

Themen

- Wartungsmaßnahmen für Amazon DocumentDB
- Versionsnummerierung der Engine
- Verwaltung Ihrer Amazon DocumentDB-Wartungsfenster
- Benachrichtigungen für Amazon DocumentDB-Engine-Patches
- Ausstehende Wartungsmaßnahmen für Amazon DocumentDB anzeigen
- Aktualisierungen der Amazon DocumentDB-Engine
- Unterversion-Upgrades
- Aktualisierungen des Amazon DocumentDB-Betriebssystems
- User-initiated aktualisiert
- Patchen globaler Cluster

Wartungsmaßnahmen für Amazon DocumentDB

Die folgenden Wartungsmaßnahmen gelten für Amazon DocumentDB-Cluster:

- `system-update`— Aktualisieren Sie den Engine-Patch für den Amazon DocumentDB-Cluster. Weitere Informationen finden Sie unter Aktualisierungen der Amazon DocumentDB-Engine.
- `os-upgrade`— Aktualisieren Sie die Betriebssysteme aller DB-Instances im Amazon DocumentDB-Cluster mithilfe fortlaufender Upgrades. Weitere Informationen finden Sie unter Aktualisierungen des Amazon DocumentDB-Betriebssystems.

Die folgenden Wartungsmaßnahmen gelten für Amazon DocumentDB-Instances:

- `system-update`— Führen Sie ein Upgrade des Betriebssystems der Amazon DocumentDB-Instance durch. Wir empfehlen, stattdessen die `os-upgrade` Wartungsaktion auf Cluster-Ebene zu verwenden. Weitere Informationen finden Sie unter Aktualisierungen des Amazon DocumentDB-Betriebssystems.

Versionsnummerierung der Engine

Amazon DocumentDB verwendet zwei separate Versionskennungen:

- Engine-Version — eine dreiteilige Nummer in der Form *`major`*`.`*`major`*`.`*`minor` (z. B. `5.0.0` oder). `5.0.1` Die ersten beiden Teile (`5.0`) sind die MongoDB-Kompatibilitätsversion; der dritte Teil ist die Nebenversion, die erhöht wird, wenn Amazon DocumentDB eine Nebenversion veröffentlicht, die Bugfixes und wichtige Verbesserungen enthält. Dies ist die Version, die Sie beim Erstellen oder Aktualisieren eines Clusters angeben.
- Engine-Patch-Version — eine separate dreiteilige Zahl in der Form *`major`*`.0.`*`patch` (z. B.`3.0.17983`), die das auf Ihren Cluster angewendete Patch-Level identifiziert. Die mittlere Ziffer ist immer`0`. Patch-Versionen enthalten wichtige Sicherheits- und Stabilitätsverbesserungen.

Wenn die Engine-Patch-Version eigenständig erscheint, können Sie die Engine-Version anhand ihres Präfixes ermitteln, wie in der folgenden Tabelle dargestellt.

Versionspräfix für den Engine-Patch	Version der Amazon DocumentDB-Engine
`1.0.`*`x`*	3.6
`2.0.`*`x`*	4,0
`3.0.`*`x`*	5.0
`4.0.`*`x`*	8.0

Um die Patch-Version zu überprüfen, auf der Ihr Cluster ausgeführt wird, stellen Sie eine Verbindung her und führen `db.runCommand({getEngineVersion: 1})` Sie ihn aus.

> **Note**
>
> `db.runCommand({getEngineVersion: 1})`Gibt ab Amazon DocumentDB 5.0.2 und 8.0.2 die Engine-Version und die Engine-Patch-Version als einzelne Zeichenfolge im Formular zurück. *`engine-version`*+*`engine-patch-version`* Beispielsweise kehrt ein Cluster auf Amazon DocumentDB 5.0.2 zurück`5.0.2+3.0.12345`, und ein Cluster auf 8.0.2 kehrt zurück. `8.0.2+4.0.12345` Cluster in früheren Versionen geben die Engine-Patch-Version eigenständig zurück, z. B. `3.0.12345`
> Wenn Sie über Skripts oder Monitore verfügen, die die Ausgabe von analysieren`getEngineVersion`, aktualisieren Sie sie, damit sie das kombinierte Format verarbeiten.

Eine Liste der veröffentlichten Engine-Patch-Versionen und deren Inhalt finden Sie unter Versionshinweise.

Verwaltung Ihrer Amazon DocumentDB-Wartungsfenster

Jeder Cluster und jede Instance hat ihr eigenes wöchentliches Wartungsfenster von 30 Minuten — den Zeitraum, in dem geplante Änderungen und Software-Patches ausgeführt werden. Die meisten Ereignisse werden innerhalb von 30 Minuten abgeschlossen; größere Ereignisse können länger dauern.

Wenn Sie bei der Erstellung der Ressource kein Zeitfenster wählen, weist Amazon DocumentDB innerhalb eines für die Region definierten 8-stündigen Tagesblocks an einem zufällig ausgewählten Tag eines nach dem Zufallsprinzip zu. Wählen Sie Fenster, die die Auswirkungen auf Ihre Anwendung so gering wie möglich halten, z. B. abends oder am Wochenende.

Für Upgrades der Datenbank-Engine verwendet Amazon DocumentDB das Fenster des Clusters, nicht die Fenster einzelner Instances.

Die folgende Tabelle zeigt die Standard-Zeitblöcke pro Region.

Name der Region	Region	UTC-Zeitblock
USA Ost (Ohio)	us-east-2	03:00-11:00
USA Ost (Nord-Virginia)	us-east-1	03:00-11:00

Name der Region	Region	UTC-Zeitblock
USA West (Oregon)	us-west-2	06:00-14:00
Afrika (Kapstadt)	af-south-1	03:00 — 11:00
Asien-Pazifik (Hongkong)	ap-east-1	06:00-14:00
Asien-Pazifik (Hyderabad)	ap-south-2	06:30 — 14:30
Asien-Pazifik (Malaysia)	ap-southeast-5	13:00-21:00
Asien-Pazifik (Mumbai)	ap-south-1	06:00-14:00
Asia Pacific (Osaka)	ap-northeast-3	12:00-20:00
Asien-Pazifik (Seoul)	ap-northeast-2	13:00-21:00
Asien-Pazifik (Singapur)	ap-southeast-1	14:00-22:00
Asien-Pazifik (Sydney)	ap-southeast-2	12:00-20:00
Asien-Pazifik (Jakarta)	ap-southeast-3	08:00-16:00
Asien-Pazifik (Melbourne)	ap-southeast-4	11:00-19:00
Asien-Pazifik (Thailand)	ap-southeast-7	15:00-23:00
Asien-Pazifik (Tokio)	ap-northeast-1	13:00-21:00
Kanada (Zentral)	ca-central-1	03:00-11:00
Kanada West (Calgary)	ca-west-1	18:00-02:00
China (Beijing)	cn-north-1	06:00-14:00
China (Ningxia)	cn-northwest-1	06:00-14:00
Europa (Frankfurt)	eu-central-1	21:00-05:00
Europa (Zürich)	eu-central-2	02:00-10:00

Name der Region	Region	UTC-Zeitblock
Europa (Irland)	eu-west-1	22:00-06:00
Europa (London)	eu-west-2	22:00-06:00
Europa (Milan)	eu-south-1	02:00-10:00
Europa (Paris)	eu-west-3	23:59-07:29
Europa (Spain)	eu-south-2	02:00 — 10:00
Europa (Stockholm)	eu-north-1	04:00 — 12:00
Mexiko (Zentral)	mx-central-1	03:00-11:00
Naher Osten (VAE)	me-central-1	05:00 — 13:00
Südamerika (São Paulo)	sa-east-1	00:00-08:00
Israel (Tel Aviv)	il-central-1	04:00-12:00
AWS GovCloud (US-East)	us-gov-east-1	17:00-01:00
AWS GovCloud (US-West)	us-gov-west-1	06:00-14:00

Ändern Sie Ihre Amazon DocumentDB-Wartungsfenster

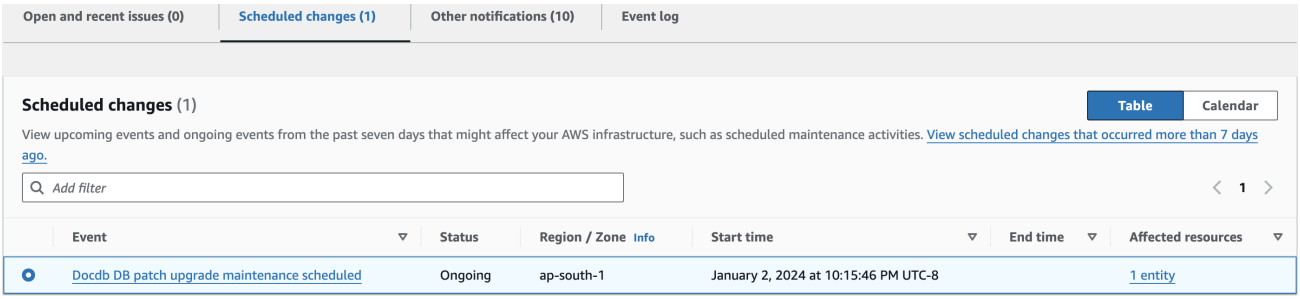
Wählen Sie das Fenster mit dem niedrigsten Verkehrsaufkommen, das Sie können, und passen Sie es im Laufe der Zeit an, wenn sich Ihre Verkehrsmuster ändern. Der Cluster oder die Instance ist während des Zeitfensters nur dann nicht verfügbar, wenn eine Systemänderung — z. B. ein Scale-Storage-Vorgang oder eine Änderung der Instance-Klasse — einen Ausfall erfordert, und zwar nur so lange, wie diese Änderung tatsächlich erforderlich ist.

So ändern Sie das Wartungsfenster

- Für einen Cluster: Weitere Informationen finden Sie unter Ändern eines Amazon DocumentDB-Clusters.
- Für eine Instance: Weitere Informationen finden Sie unter Ändern einer Amazon DocumentDB DocumentDB-Instance.

Benachrichtigungen für Amazon DocumentDB-Engine-Patches

Wenn ein erforderlicher Engine-Patch in einer AWS Region verfügbar wird, erhält jedes AWS Konto mit einem betroffenen Amazon DocumentDB-Cluster in dieser Region eine Benachrichtigung über das Health Dashboard (AHD) und per E-Mail (gesendet an die Root-Benutzeradresse des AWS Kontos). Eine Benachrichtigung wird pro betroffener Amazon DocumentDB-Engine-Version zugestellt. Sie finden sie im AHD unter Geplante Änderungen. In jeder Benachrichtigung sind der Zeitpunkt der Patch-Verfügbarkeit, der Zeitplan für die automatische Anwendung, die betroffenen Cluster und die Versionshinweise aufgeführt.



Für erforderliche Engine-Patches gilt eine einzige Vorlaufzeit von etwa 30 Tagen. Wenn ein Patch in Ihrer Region verfügbar ist, sendet Amazon DocumentDB die oben beschriebene Benachrichtigung. Zu diesem Zeitpunkt `AutoAppliedAfterDate` ist der Patch auf etwa 30 Tage später festgelegt. Bis zu diesem Datum steht der Patch noch aus: Sie können ihn jederzeit anwenden oder ihn verschieben, indem Sie das Wartungsfenster Ihres Clusters auf einen späteren Tag verschieben. An oder nach dem wird der `AutoAppliedAfterDate` Patch während des nächsten Wartungsfensters für den Cluster automatisch angewendet.

Beispiel: Ein erforderlicher Patch, der am 1. Juni 2026 verfügbar wird, hat einen Gültigkeitszeitraum `AutoAppliedAfterDate` von ungefähr dem 1. Juli 2026. Sie erhalten die Benachrichtigung am 1. Juni 2026, und wenn Sie keine Maßnahmen ergreifen, wird der Patch während des ersten Wartungsfensters Ihres Clusters am oder nach dem 1. Juli 2026 automatisch angewendet.

Nach Erhalt der Benachrichtigung haben Sie zwei Möglichkeiten: Sie können den Patch vor dem Datum der automatischen Anwendung selbst anwenden oder warten, bis er während eines bevorstehenden Wartungsfensters automatisch angewendet wird (Standardeinstellung). Um sich selbst zu installieren, öffnen Sie die Registerkarte Wartung und Backups des Clusters und suchen Sie nach dem entsprechenden Eintrag. `system-update`

> **Note**
>
> Der Status der Benachrichtigung im AHD bleibt so lange erhalten, bis Amazon DocumentDB einen weiteren Engine-Patch mit einer neuen Patch-Version veröffentlicht. Nachdem der Patch angewendet wurde, wird die Engine-Patch-Version des Clusters aktualisiert, sodass sie mit der Version in der Benachrichtigung übereinstimmt. Überprüfen Sie die neue Version, indem Sie Folgendes ausführen`db.runCommand({getEngineVersion: 1})`.

Optionale Patches und neue Nebenversionen generieren keine AHD- oder E-Mail-Benachrichtigungen. Um sie zu verfolgen, schauen Sie sich die Amazon DocumentDB-Versionshinweise an.

Erzwungene Patches (die seltenste Kategorie, die den kritischsten Sicherheitsupdates vorbehalten ist) werden ebenfalls per AHD und E-Mail angekündigt. Im Gegensatz zu erforderlichen Patches werden sie außerhalb Ihres Wartungsfensters angewendet, sodass das obige Beispiel für die automatische Anwendung nicht zutrifft.

Programmgesteuertes Reagieren auf Patch-Benachrichtigungen

AWS Health ist in Amazon integriert EventBridge, sodass Sie ereignisgesteuerte Anwendungen für mehr als 20 Ziele erstellen können, einschließlich AWS Lambda Amazon Simple Queue Service (SQS). Um programmgesteuert auf die Verfügbarkeit von Engine-Patches zu reagieren, konfigurieren Sie die Konfiguration entsprechend dem Ereignis. EventBridge `AWS_DOCDB_DB_PATCH_UPGRADE_MAINTENANCE_SCHEDULED` Von dort aus können Sie Eventdaten erfassen, zusätzliche Ereignisse auslösen, Push-Benachrichtigungen über das versenden oder jede andere Aktion ergreifen AWS Console Mobile Application, die Sie benötigen.

Wenn Amazon DocumentDB einen Patch storniert (selten), erhalten Sie eine AHD-Benachrichtigung und eine E-Mail über die Kündigung. Verwenden Sie den `AWS_DOCDB_DB_PATCH_UPGRADE_MAINTENANCE_CANCELLED` Eventcode bei Amazon EventBridge , um diesen Fall zu bearbeiten. Weitere Informationen zum Schreiben von Regeln finden Sie im EventBridge Amazon-Benutzerhandbuch.

Ausstehende Wartungsmaßnahmen für Amazon DocumentDB anzeigen

Verwenden Sie AWS-Managementkonsole oder die AWS CLI , um zu überprüfen, welche Wartungsarbeiten für einen Cluster oder eine Instance ausstehen.

Ausstehende Updates werden mit dem Aktionstyp angezeigt`system-update`, der sowohl Engine-Patches als auch Betriebssystemupdates abdeckt.

Wenn ein Update aussteht, können Sie:

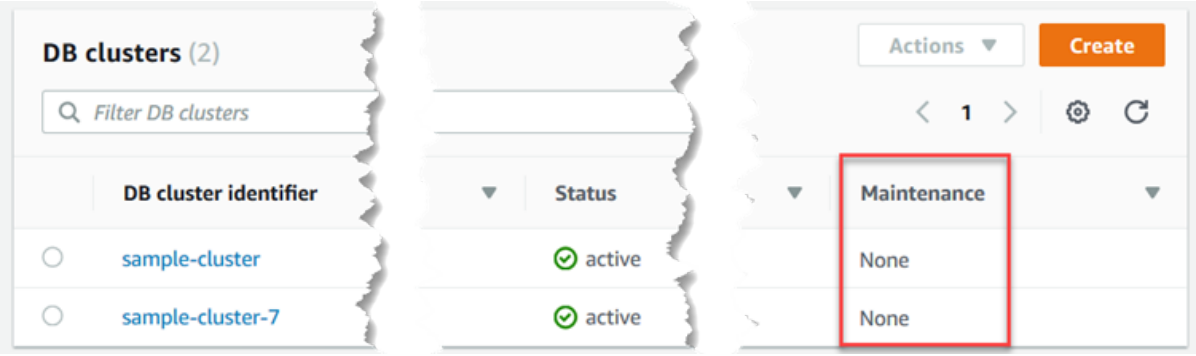
- Wenden Sie es sofort an.
- Planen Sie es für das nächste Wartungsfenster ein.
- Verschieben Sie es (nur Engine-Patches und Betriebssystem-Updates), indem Sie vorher `AutoAppliedAfterDate` Ihr Wartungsfenster ändern. Sobald dieses Datum abgelaufen ist, wird die Aktion im nächsten Wartungsfenster automatisch angewendet. Nach Ablauf `ForcedApplyDate` dieser Frist ist kein weiterer Aufschub mehr möglich.

> **Note**
>
> Wenn Sie keine Maßnahmen ergreifen, werden erforderliche Wartungsmaßnahmen wie z. B. erforderliche Engine-Patches während eines bevorstehenden Wartungsfensters automatisch angewendet. Optionale Patches und Nebenversionen werden niemals automatisch angewendet.

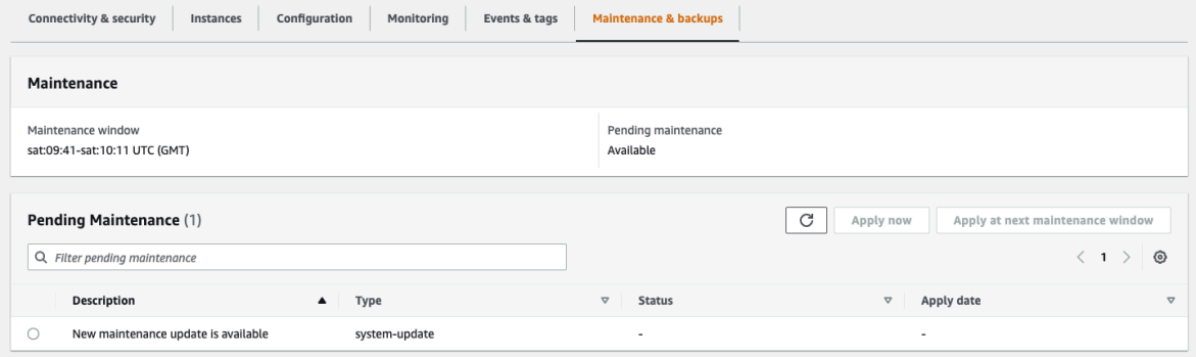
Das Wartungsfenster steuert, wann ausstehende Operationen beginnen, nicht, wie lange sie dauern, bis sie abgeschlossen sind.

Using the AWS-Managementkonsole

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Klicken Sie im Navigationsbereich auf Cluster.
3. In der Spalte „Wartung“ des Clusters wird „Verfügbar“, „Erforderlich“ oder „Nächstes Fenster“ angezeigt, wenn ein Update aussteht.



4. Öffnen Sie den Cluster und wählen Sie dann Wartung und Backups aus, um die Elemente für ausstehende Wartungsarbeiten anzuzeigen und entsprechend zu handeln.



Using the AWS CLI

Führen Sie aus`describe-pending-maintenance-actions`, um zu sehen, was noch aussteht. Das folgende Beispiel zeigt ein Konto ohne ausstehende Aktionen.

```
aws docdb describe-pending-maintenance-actions
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    "PendingMaintenanceActions": []
}
```

Ein Konto mit einer ausstehenden Aktion gibt eine Ausgabe zurück, die wie folgt aussieht:

```
{
    "PendingMaintenanceActions": [
        {
```

```
            "ResourceIdentifier": "arn:aws:rds:us-
east-1:123456789012:cluster:sample-cluster",
            "PendingMaintenanceActionDetails": [
                {
                    "Action": "system-update",
                    "Description": "db-version-upgrade",
                    "CurrentApplyDate": "2026-05-15T03:01:00Z",
                    "AutoAppliedAfterDate": "2026-05-15T03:01:00Z"
                }
            ]
        }
    ]
}
```

Sie können die Liste auf bestimmte Cluster beschränken`--filters`, indem Sie das Formular verwenden`Name=`*`filter-name`*`,Values=`*`resource-id`*`,...`. Der akzeptierte Filter `Name` ist`db-cluster-id`, der eine Liste von Cluster-Identifikatoren oder ARNs verwendet.

Example

Für Linux, macOS oder Unix:

```
aws docdb describe-pending-maintenance-actions \
   --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

Für Windows:

```
aws docdb describe-pending-maintenance-actions ^
   --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

Termine anwenden

Für jede ausstehende Wartungsmaßnahme gelten bis zu drei Anwendungsdaten. Sie erscheinen in der AWS CLI Ausgabe für `describe-pending-maintenance-actions` und geben an, wann die Aktion ausgeführt wird. Felder sind `null` für die optionale Wartung vorgesehen.

- **`CurrentApplyDate`**— wann die Aktion geplant ist, entweder jetzt oder im nächsten Wartungsfenster. Wird für erforderliche und erzwungene Aktionen ausgefüllt.

- **`AutoAppliedAfterDate`**— das Datum, nach dem die automatische Anwendung während des Wartungsfensters für den Cluster oder die Instance beginnt. Wird für die erforderlichen Aktionen ausgefüllt.
- **`ForcedApplyDate`**— die harte Frist. Nach diesem Datum wird die Aktion unabhängig von Ihrem Wartungsfenster automatisch ausgeführt. Wird für erzwungene Aktionen ausgefüllt.

Um eine ausstehende Aktion aufzuschieben, verschieben Sie Ihr Wartungsfenster auf einen späteren Tag zuvor`AutoAppliedAfterDate`. Sobald der `AutoAppliedAfterDate` Vorgang abgeschlossen ist, wird die Aktion im nächsten Wartungsfenster automatisch angewendet. Sobald der `ForcedApplyDate` Vorgang abgeschlossen ist, ist kein weiterer Aufschub mehr möglich. Das genaue Zeitfenster für den Aufschub ist je nach Patch unterschiedlich. Die Daten werden in der AHD-Benachrichtigung und in der Ausgabe veröffentlicht. AWS CLI

Aktualisierungen der Amazon DocumentDB-Engine

Wenn Sie einen ausstehenden Engine-Patch identifiziert haben, wenden Sie eines der folgenden Verfahren an, um ihn anzuwenden oder zu planen. Sie können diese Prozeduren entweder von AWS-Managementkonsole oder von ausführen AWS CLI.

Using the AWS-Managementkonsole

Um ein Update für einen Cluster zu verwalten

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Klicken Sie im Navigationsbereich auf Cluster.
3. Wählen Sie den Cluster aus, den Sie aktualisieren möchten.
4. Wählen Sie im Menü „Aktionen“ eine der folgenden Optionen aus:
   - Jetzt aktualisieren — führen Sie die ausstehenden Wartungsarbeiten sofort aus.
   - Upgrade im nächsten Fenster — Führen Sie das Upgrade während des nächsten Wartungsfensters des Clusters aus.

   Sie können auch „Jetzt anwenden“ oder „Im nächsten Wartungsfenster anwenden“ im Abschnitt „Ausstehende Wartung“ auf der Registerkarte „Wartung und Backups“ des Clusters verwenden (sieheAusstehende Wartungsmaßnahmen für Amazon DocumentDB anzeigen).

> **Note**
>
> Wenn nichts aussteht, sind alle diese Optionen inaktiv.

Using the AWS CLI

Wenden Sie ein ausstehendes Update mit an`apply-pending-maintenance-action`.

Parameters

- **`--resource-identifier`**— Der Amazon DocumentDB Amazon-Ressourcenname (ARN) der Ressource, auf die die ausstehende Aktion abzielt.
- **`--apply-action`**— die ausstehende Wartungsmaßnahme, die angewendet werden soll. Wird verwendet`system-update`, um einen Engine-Patch anzuwenden.
- **`--opt-in-type`**— die Art der Opt-in-Anfrage oder ob eine solche rückgängig gemacht werden soll. Zulässige Werte:
  - `immediate`— bewirb dich jetzt. Kann nach dem Absenden nicht rückgängig gemacht werden.
  - `next-maintenance`— gilt während des nächsten Wartungsfensters der Ressource.
  - `undo-opt-in`— ein bestehendes `next-maintenance` Opt-In stornieren.

Example

Für Linux, macOS oder Unix:

```
aws docdb apply-pending-maintenance-action \
    --resource-identifier arn:aws:rds:us-east-1:123456789012:db:sample-cluster-instance-1 \
    --apply-action system-update \
    --opt-in-type immediate
```

Für Windows:

```
aws docdb apply-pending-maintenance-action ^
    --resource-identifier arn:aws:rds:us-east-1:123456789012:db:sample-cluster-instance-1 ^
    --apply-action system-update ^
```

```
    --opt-in-type immediate
```

Lesen Sie die Verfügbarkeit beim Patchen

Die Amazon DocumentDB-Engine 5.0 und 8.0 behalten die Leseverfügbarkeit beim Patchen bei, wenn der Cluster mehrere Instances hat. Amazon DocumentDB patcht die Reader-Instances fortlaufend in drei Gruppen, sodass die verbleibenden Reader weiterhin den Datenverkehr verarbeiten. Der Writer ist während der Patches kurzzeitig nicht verfügbar. Um keine Ausfallzeiten beim Lesen zu erreichen, legen Sie Ihre Leseeinstellungen so fest, dass Lesevorgänge auf den Writer zurückgreifen können: `secondaryPreferred` oder `primaryPreferred` funktionieren; `primary` oder `secondary` allein kann es zu Leseausfällen kommen.

Modus „Lesepräferenz“	Während des Writer-Upgrades	Während des Reader-Upgrades	Erforderliche Mindestanzahl an Lesegeräten, um Ausfallzeiten beim Lesen zu vermeiden
`primary`	Read/write Ausfallzeiten	Keine Auswirkungen	N/A
`primaryPreferred`	Schreiben Sie Ausfallzeiten	Keine Auswirkungen	1
`secondary`	Schreiben Sie Ausfallzeiten	Ausfallzeiten lesen (wenn nur ein Leser vorhanden ist)	2
`secondaryPreferred`	Ausfallzeiten schreiben	Keine Auswirkungen	1
`nearest`	Schreiben Sie Ausfallzeiten	Keine Auswirkungen	1

Während die Leser patchen, sinkt der gesamte Cluster-Lesedurchsatz vorübergehend. Um den Durchsatz konstant zu halten, sollten Sie vor dem Upgrade zusätzliche Lesegeräte bereitstellen und sie nach Abschluss des Upgrades entfernen.

Auf Engine 3.6 und 4.0 gelten diese Funktionen zur Leseverfügbarkeit nicht: Ein Engine-Patch verursacht längere Ausfallzeiten, die sich sowohl auf Lese- als auch auf Schreibvorgänge auswirken. Informationen zum Upgrade auf eine Hauptversion, bei der dies der Fall ist, finden Sie unter. Direktes Upgrade der Hauptversion von Amazon DocumentDB

Dauer der Patch-Ausfallzeit

Engine-patch Die Ausfallzeiten variieren. Die wichtigsten Faktoren sind die CPU-Auslastung und die Speicherbelastung der Instance zum Zeitpunkt des Patches. Daher ist die richtige Dimensionierung Ihrer Instances von entscheidender Bedeutung. Um Ausfallzeiten zu minimieren, führen Sie die neueste Version der Amazon DocumentDB-Haupt-Engine aus und verteilen Sie die Instances auf mehrere Availability Zones.

Patch-Updates und -Ersetzungen

Amazon DocumentDB überwacht Patches nach der Veröffentlichung. In dem seltenen Fall, dass ein Problem erkannt wird, unterbricht Amazon DocumentDB den Rollout, während eine aktualisierte Version vorbereitet wird. In diesem Fall sehen Cluster, die den Patch noch nicht erhalten haben, ihn nicht mehr als verfügbare Wartungsmaßnahme an, und die entsprechende Benachrichtigung über geplante Änderungen in der wird zurückgezogen. Health Dashboard Cluster, auf denen die betroffene Version bereits ausgeführt wird, funktionieren weiterhin normal und erfordern keine Maßnahmen Ihrerseits.

Ein aktualisierter Patch folgt in Kürze. Sobald es in Ihrer Region verfügbar ist, erhalten Sie eine neue Benachrichtigung per E-Mail Health Dashboard und, wie unter beschriebenBenachrichtigungen für Amazon DocumentDB-Engine-Patches.

Unterversion-Upgrades

Amazon DocumentDB veröffentlicht Nebenversionen zusätzlich zur Hauptversion 5.0 und höher (z. B.`5.0.1`). Für Hauptversionen vor 5.0 werden keine Nebenversionen veröffentlicht. Nebenversionen verhalten sich anders als erforderliche und optionale Engine-Patches:

- Sie werden nicht als ausstehende Wartungsmaßnahme angezeigt und werden niemals automatisch angewendet.
- Sie generieren keine AHD- oder E-Mail-Benachrichtigungen. Neue Nebenversionen werden in den Amazon DocumentDB-Versionshinweisen angekündigt.
- Für ein Upgrade ändern Sie die Engine-Version des Clusters (sofort oder während des nächsten Wartungsfensters). Upgrades von Nebenversionen erfordern eine kurze Ausfallzeit und sind

einseitig — Sie können kein Downgrade auf eine frühere Nebenversion durchführen. Führen Sie bei globalen Clustern ein Upgrade der sekundären Cluster vor den primären Clustern durch.

Lesen Sie mehr:Upgrade der Nebenversion von Amazon DocumentDB.

Aktualisierungen des Amazon DocumentDB-Betriebssystems

Instances benötigen gelegentlich Betriebssystem-Updates. Amazon DocumentDB aktualisiert das Betriebssystem, um die Leistung zu verbessern und die Sicherheit zu erhöhen. Bei Betriebssystemaktualisierungen bleiben die Cluster-Engine-Version und die Instance-Klasse unverändert. Wie Engine-Patches verwenden Betriebssystemupdates den oben in diesem Thema beschriebenen optionalen /erforderlich/erzwungenen Lebenszyklus. Im Gegensatz zu Engine-Patches kann ein Betriebssystem-Update diese Kategorien im Laufe der Zeit durchgehen, wenn Sie es verschieben. Wenden Sie Betriebssystemupdates an, sobald sie verfügbar sind, und legen Sie die Wartungsfenster für Cluster und Instances auf Zeiten fest, die Ihren Geschäftsanforderungen entsprechen.

Verwenden Sie die `os-upgrade` Wartungsaktion auf Cluster-Ebene, um Betriebssystem-Updates auf alle Instances in einem Cluster anzuwenden. Amazon DocumentDB aktualisiert Instances fortlaufend, einige nacheinander, und aktualisiert die primäre Instance zuletzt, um Failovers zu minimieren. Das Update wird während des Cluster-Wartungsfensters ausgeführt — nicht während des Wartungsfensters für einzelne Instances —, das Sie konfigurieren.

Nachdem eine Instanz ein Betriebssystem-Update erhalten hat, beginnt ihr Puffercache leer. Bis das Working Set wieder aus dem Speichervolume aufgefüllt ist, kann es bei Abfragen auf dieser Instance zu einer höheren und niedrigeren `BufferCacheHitRatio` Latenz kommen.

Wenn Amazon DocumentDB die primäre Instance aktualisiert, stuft ein Failover ein Replikat zur neuen primären Instance herauf. Verwenden Sie den Cluster-Endpunkt, damit Ihre Anwendung dies transparent handhabt. Um die Leseverfügbarkeit während der Aktualisierung von Instances aufrechtzuerhalten, setzen Sie Ihre Lesepräferenz auf `secondaryPreferred` oder, `primaryPreferred` sodass Lesevorgänge auf eine verfügbare Instance zurückgreifen können. Halten Sie potenzielle Failover-Ziele (Replikate mit der höchsten Prioritätsstufe) in derselben Instance-Klasse wie die primäre Instance-Klasse. Dadurch wird eine Verschlechterung der Schreibleistung nach einer Heraufstufung vermieden. Details hierzu finden Sie unter Amazon DocumentDB-Failover.

Sowohl die Aktionen auf Clusterebene `os-upgrade` als auch auf Instanzebene werden möglicherweise gleichzeitig als `system-update` verfügbare Aktionen angezeigt. `describe-`

`pending-maintenance-actions` Sie können jedoch nicht beide gleichzeitig planen. Wenn `system-update` Aktionen auf Instanzebene auf einer Instance aktiv geplant sind, müssen Sie sie abbrechen oder abschließen, bevor Sie die `os-upgrade` Aktion auf Clusterebene planen, und umgekehrt.

> **Important**
>
> Ihre Amazon DocumentDB-Instance geht für das Betriebssystem-Update offline. Multi-instance Cluster minimieren die Auswirkungen. Wenn Sie einen Single-Instance-Cluster ausführen, können Sie vorübergehend einen sekundären Cluster für das Update hinzufügen und ihn anschließend entfernen. Für den Sekundärserver fallen die üblichen Gebühren an, solange er existiert.

> **Note**
>
> Die `system-update` Aktion auf Instanzebene ist aus Gründen der Abwärtskompatibilität weiterhin verfügbar. Wenn Sie sie verwenden müssen, aktualisieren Sie zuerst die Replikate und zuletzt die Primärreplikate. Vermeiden Sie es, sie gleichzeitig zu patchen, da ein Failover während des Patches zu längeren Ausfallzeiten führen kann.

Wenn Sie über ein Ereignis informiert werden möchten, wenn ein neues optionales Betriebssystemupdate eintrifft, melden Sie sich `RDS-EVENT-0230` in der Event-Kategorie für Sicherheitspatches an. Weitere Informationen finden Sie unter Amazon DocumentDB DocumentDB-Veranstaltungen abonnieren.

> **Note**
>
> Aus Compliance-Gründen kann es erforderlich sein, über optionale und erforderliche Updates auf dem Laufenden zu bleiben. Wenden Sie während Ihrer Wartungszeiträume routinemäßig `os-upgrade` Maßnahmen an.

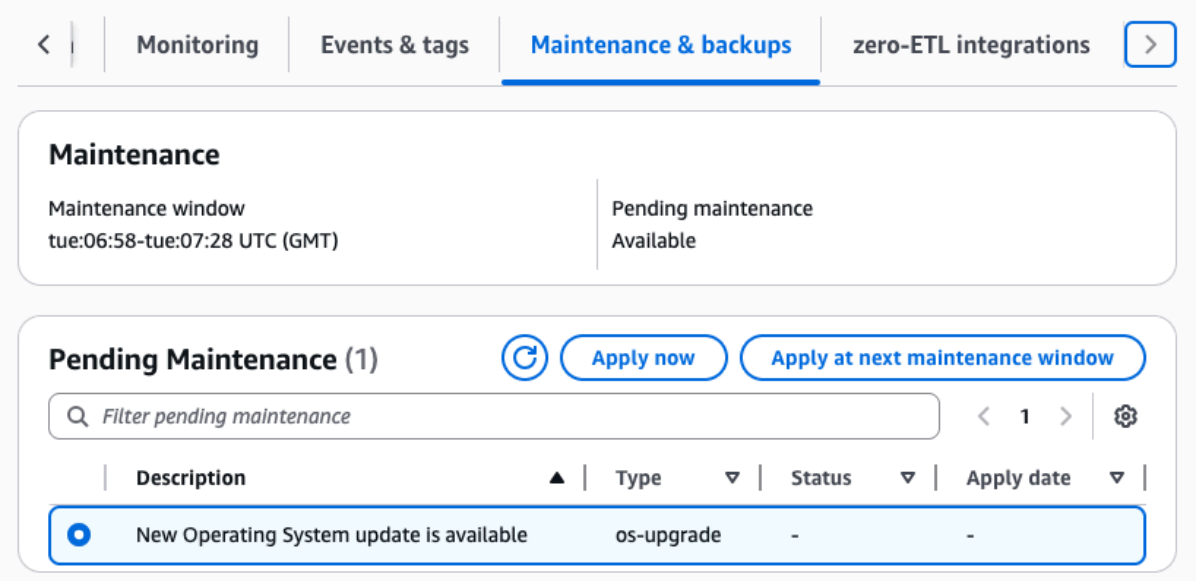
Betriebssystem-Updates sind an bestimmte Instance-Klassen gebunden, sodass verschiedene Instances zu unterschiedlichen Zeiten in Frage kommen. Wenn Ihr Cluster nicht über den neuesten Engine-Patch verfügt, wird das Betriebssystem-Update möglicherweise nicht angezeigt. Wenden Sie zuerst den neuesten Engine-Patch an (siheAktualisierungen der Amazon DocumentDB-Engine).

Verwenden Sie AWS-Managementkonsole oder AWS CLI , um zu überprüfen, ob ein Update verfügbar ist.

Using the AWS-Managementkonsole

Um von der Konsole aus nach einem Betriebssystem-Update zu suchen:

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Wählen Sie im Navigationsbereich Clusters und dann den Cluster-Namen aus.
3. Wählen Sie die Registerkarte Wartung und Backups.
4. Unter Ausstehende Wartung wird die **`os-upgrade`** Aktion angezeigt, wenn ein Betriebssystem-Update verfügbar ist.



5. Wählen Sie die `os-upgrade` Aktion aus und wählen Sie Jetzt anwenden oder Im nächsten Wartungsfenster anwenden. Wenn der Wert nächstes Fenster ist, können Sie die Aktualisierung mit „Upgrade aufschieben“ verschieben, solange die Aktion noch nicht gestartet wurde.

Using the AWS CLI

Suchen Sie nach einem ausstehenden Betriebssystem-Update:

```
aws docdb describe-pending-maintenance-actions
```

```
{
    "PendingMaintenanceActions": [
        {
            "ResourceIdentifier": "arn:aws:rds:aa-
example-1:111122223333:cluster:sample-cluster",
            "PendingMaintenanceActionDetails": [
                {
                    "Action": "os-upgrade",
                    "Description": "New Operating System update is available"
                }
            ]
        },
        {
            "ResourceIdentifier": "arn:aws:rds:aa-example-1:111122223333:db:sample-
cluster-instance-1",
            "PendingMaintenanceActionDetails": [
                {
                    "Action": "system-update",
                    "Description": "New Operating System update is available"
                }
            ]
        },
        {
            "ResourceIdentifier": "arn:aws:rds:aa-example-1:111122223333:db:sample-
cluster-instance-2",
            "PendingMaintenanceActionDetails": [
                {
                    "Action": "system-update",
                    "Description": "New Operating System update is available"
                }
            ]
        }
    ]
}
```

Betriebssystemupdates werden auf Clusterebene als `os-upgrade` und auf Instanzebene als angezeigt`system-update`. Verwenden Sie die Aktion auf Clusterebene. `os-upgrade`

Example

Im folgenden Beispiel wird das Betriebssystem-Update sofort angewendet.

Für Linux, macOS oder Unix:

```
aws docdb apply-pending-maintenance-action \
    --resource-identifier arn:aws:rds:aa-example-1:111122223333:cluster:sample-
cluster \
    --apply-action os-upgrade \
    --opt-in-type immediate
```

Für Windows:

```
aws docdb apply-pending-maintenance-action ^
    --resource-identifier arn:aws:rds:aa-example-1:111122223333:cluster:sample-
cluster ^
    --apply-action os-upgrade ^
    --opt-in-type immediate
```

User-initiated aktualisiert

Einige Änderungen beginnen Sie selbst, z. B. indem Sie eine Instanzklasse gegen eine mit mehr oder weniger Arbeitsspeicher austauschen oder die Parametergruppe des Clusters ändern. Amazon DocumentDB behandelt diese anders als Aktualisierungen, die es initiiert. Details hierzu finden Sie unter:

- Ändern eines Amazon DocumentDB-Clusters
- Ändern einer Amazon DocumentDB DocumentDB-Instance

Um vom Benutzer initiierte Änderungen aufzulisten, die noch ausstehen:

Example

Um ausstehende benutzerinitiierte Änderungen für Ihre Instances aufzulisten

Für Linux, macOS oder Unix:

```
aws docdb describe-db-instances \
    --query 'DBInstances[*].
[DBClusterIdentifier,DBInstanceIdentifier,PendingModifiedValues]'
```

Für Windows:

```
aws docdb describe-db-instances ^
```

```
    --query 'DBInstances[*].
[DBClusterIdentifier,DBInstanceIdentifier,PendingModifiedValues]'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

In diesem Beispiel `sample-cluster-instance` hat eine ausstehende Änderung zu`db.r5.xlarge`; `sample-cluster-instance-2` hat keine.

```
[
    [
        "sample-cluster",
        "sample-cluster-instance",
        {
            "DBInstanceClass": "db.r5.xlarge"
        }
    ],
    [
        "sample-cluster",
        "sample-cluster-instance-2",
        {}
    ]
]
```

Patchen globaler Cluster

In einem globalen Cluster wird jeder Mitgliedscluster — primärer und sekundärer — während seines eigenen Wartungsfensters aktualisiert. Wenn ein erforderlicher Engine-Patch in jeder Region verfügbar ist, erhalten Sie eine AHD- und E-Mail-Benachrichtigung. Optionale Patches und neue Nebenversionen generieren keine Benachrichtigungen. Diese finden Sie in den Amazon DocumentDB-Versionshinweisen.

Wenn Sie sich selbst bewerben, patchen Sie die sekundären Patches immer zuerst und die primäre zuletzt. Bei dieser Reihenfolge sind Failover und Switchover während des gesamten Rollouts verfügbar.

> **Important**
>
> Wenn Sie versehentlich zuerst die primäre Version patchen, bringen Sie alle sekundären Geräte so schnell wie möglich auf dieselbe Version. Failover und Switchover bleiben deaktiviert, bis alle Cluster dieselbe Version verwenden.

Wenn Sie keine Maßnahmen ergreifen, wird der Patch im nächsten Wartungsfenster jedes Clusters automatisch angewendet: zuerst die sekundären, dann der primäre in seinem Fenster, sobald die sekundären Wartungsarbeiten abgeschlossen sind.

Behalten Sie für den primären und den sekundären DB-Cluster dieselbe Version bei. Ein verwaltetes regionsübergreifendes Failover funktioniert nur in einer globalen Datenbank, wenn alle Cluster dieselbe Engine-Version und dieselbe Patch-Level verwenden. Das Gleiche gilt, wenn Sie eine neue sekundäre Engine-Version hinzufügen, die eine neuere Engine-Version als die primäre Version verwendet. Erstellen Sie neue sekundäre Engines in der Version der primären Engine, bevor Sie sie der globalen Datenbank hinzufügen.

Führen Sie nach einer Patch-Benachrichtigung zum frühestmöglichen Zeitpunkt ein Upgrade der primären und sekundären Version auf die neueste Version durch, damit Failover und Switchover weiterhin funktionieren. Wenn eine Failover- oder Switchover-Anfrage abgelehnt wird, vergleichen Sie die Engine-Patch-Versionen der Cluster. Wenn sie nicht übereinstimmen, wenden Sie den verfügbaren Patch auf die verzögerten Cluster an.

Grundlegendes zu serviceverknüpften Rollen

Amazon DocumentDB (mit MongoDB-Kompatibilität) verwendet AWS Identity and Access Management (IAM) serviceverknüpfte Rollen. Eine serviceverknüpfte Rolle ist eine einzigartige Art von IAM-Rolle, die direkt mit Amazon DocumentDB verknüpft ist. Service-linked Rollen werden von Amazon DocumentDB vordefiniert und beinhalten alle Berechtigungen, die der Service benötigt, um andere AWS Dienste in Ihrem Namen aufzurufen.

Eine serviceverknüpfte Rolle erleichtert die Verwendung von Amazon DocumentDB, da Sie die erforderlichen Berechtigungen nicht manuell hinzufügen müssen. Amazon DocumentDB definiert die Berechtigungen seiner serviceverknüpften Rollen. Sofern nicht anders definiert, kann nur Amazon DocumentDB seine Rollen übernehmen. Die definierten Berechtigungen umfassen die Vertrauens- und Berechtigungsrichtlinie. Diese Berechtigungsrichtlinie kann keinen anderen IAM-Entitäten zugewiesen werden.

Sie können die Rollen nur nach dem Löschen der zugehörigen Ressourcen löschen. Dies schützt Ihre Amazon DocumentDB-Ressourcen, da Sie nicht versehentlich die Zugriffsberechtigung für die Ressourcen entziehen können.

Informationen zu anderen Diensten, die serviceverknüpfte Rollen unterstützen, finden Sie unter AWS Services, die mit IAM funktionieren. Suchen Sie nach den Diensten, für die in der Spalte Rolle

„Ja“ steht. Service-Linked Wählen Sie über einen Link Ja aus, um die Dokumentation zu einer serviceverknüpften Rolle für diesen Service anzuzeigen.

Berechtigungen für serviceverknüpfte Rollen in Amazon DocumentDB

Amazon DocumentDB (mit MongoDB-Kompatibilität) verwendet die serviceverknüpfte Rolle namens AWSServiceRoleForRDS, damit Amazon DocumentDB AWS Dienste im Namen Ihrer Cluster aufrufen kann.

Die AWSServiceRoleForRDS serviceverknüpfte Rolle vertraut darauf, dass die folgenden Dienste die Rolle übernehmen:

- `docdb.amazonaws.com`

Die Richtlinie für Rollenberechtigungen ermöglicht es Amazon DocumentDB, die folgenden Aktionen für die angegebenen Ressourcen durchzuführen:

- Aktionen auf `ec2`:
  - `AssignPrivateIpAddresses`
  - `AuthorizeSecurityGroupIngress`
  - `CreateNetworkInterface`
  - `CreateSecurityGroup`
  - `DeleteNetworkInterface`
  - `DeleteSecurityGroup`
  - `DescribeAvailabilityZones`
  - `DescribeInternetGateways`
  - `DescribeSecurityGroups`
  - `DescribeSubnets`
  - `DescribeVpcAttribute`
  - `DescribeVpcs`
  - `ModifyNetworkInterfaceAttribute`
  - `RevokeSecurityGroupIngress`
  - `UnassignPrivateIpAddresses`
- Aktionen auf `sns`:

  - `ListTopic`
  - `Publish`
- Aktionen auf `cloudwatch`:
  - `PutMetricData`
  - `GetMetricData`
  - `CreateLogStream`
  - `PullLogEvents`
  - `DescribeLogStreams`
  - `CreateLogGroup`

> **Note**
>
> Sie müssen Berechtigungen konfigurieren, damit eine juristische Stelle von IAM (z. B. Benutzer, Gruppe oder Rolle) eine serviceverknüpfte Rolle erstellen, bearbeiten oder löschen kann. Möglicherweise erhalten Sie die folgende Fehlermeldung:
> Unable to create the resource. Überprüfen Sie, ob Sie die Berechtigung haben, eine serviceverknüpfte Rolle zu erstellen. Andernfalls warten Sie und versuchen Sie es später noch einmal.
> Wenn Sie diesen Fehler erhalten, stellen Sie sicher, dass Sie die folgenden Berechtigungen aktiviert haben:
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
> AWSServiceRoleForRDS",
>     "Condition": {
>         "StringLike": {
>             "iam:AWSServiceName":"rds.amazonaws.com"
>         }
>     }
> }
> ```

Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Service-Linked Rollenberechtigungen.

Erstellen einer serviceverknüpften Amazon DocumentDB-Rolle

Sie müssen eine serviceverknüpfte Rolle nicht manuell erstellen. Wenn Sie einen Cluster erstellen, erstellt Amazon DocumentDB die serviceverknüpfte Rolle für Sie.

Wenn Sie diese serviceverknüpfte Rolle löschen und dann erneut erstellen müssen, können Sie die Rolle in Ihrem Konto mit demselben Verfahren neu anlegen. Wenn Sie einen Cluster erstellen, erstellt Amazon DocumentDB die serviceverknüpfte Rolle erneut für Sie.

Ändern einer serviceverknüpften Amazon DocumentDB-Rolle

Amazon DocumentDB erlaubt Ihnen nicht, die AWSServiceRoleForRDS serviceverknüpfte Rolle zu ändern. Da möglicherweise verschiedene Entitäten auf die Rolle verweisen, kann der Rollenname nach dem Erstellen einer serviceverknüpften Rolle nicht mehr geändert werden. Sie können die Beschreibung der Rolle jedoch mithilfe von IAM ändern. Weitere Informationen finden Sie unter Bearbeiten einer Service-Linked Rolle im IAM-Benutzerhandbuch.

Löschen einer serviceverknüpften Amazon DocumentDB-Rolle

Wenn Sie eine Funktion oder einen Dienst, für den eine serviceverknüpfte Rolle erforderlich ist, nicht mehr benötigen, löschen Sie diese Rolle. Auf diese Weise haben Sie keine ungenutzte Entität, die nicht aktiv überwacht oder verwaltet wird. Sie müssen jedoch alle Ihre -Cluster löschen, bevor Sie die serviceverknüpfte Rolle löschen können.

Bereinigen einer mit dem Service verknüpften Amazon DocumentDB-Rolle

Bevor Sie mit IAM eine serviceverknüpfte Rolle löschen können, müssen Sie sich zunächst vergewissern, dass die Rolle über keine aktiven Sitzungen verfügt, und alle Ressourcen entfernen, die von der Rolle verwendet werden.

So überprüfen Sie über die Konsole, ob die servicegebundene Rolle eine aktive Sitzung hat:

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die IAM-Konsole unter. https://console.aws.amazon.com/iam/
2. Wählen Sie im Navigationsbereich der IAM-Konsole Rollen und dann den Namen (nicht das Kontrollkästchen) der AWSServiceRoleForRDS Rolle aus.
3. Wählen Sie auf der Seite Summary für die ausgewählte Rolle die Registerkarte Access Advisor.
4. Überprüfen Sie auf der Registerkarte Access Advisor (Advisor aufrufen) die jüngsten Aktivitäten für die serviceverknüpfte Rolle.

> **Note**
>
> Wenn Sie sich nicht sicher sind, ob Amazon DocumentDB die AWSServiceRoleForRDS Rolle verwendet, können Sie versuchen, die Rolle zu löschen. Wenn der Service die Rolle verwendet, schlägt die Löschung fehl und Sie können die -Regionen anzeigen, in denen die Rolle verwendet wird. Wenn die Rolle verwendet wird, müssen Sie warten, bis die Sitzung beendet wird, bevor Sie die Rolle löschen können. Die Sitzung für eine serviceverknüpfte Rolle können Sie nicht widerrufen.

Wenn Sie die AWSServiceRoleForRDS Rolle entfernen möchten, müssen Sie zuerst alle Ihre Instances und Cluster löschen. Informationen zum Löschen von Instances und Clustern finden Sie in den folgenden Themen:

- Löschen einer Amazon DocumentDB DocumentDB-Instance
- Löschen eines Amazon DocumentDB-Clusters

Unterstützte Regionen für serviceverknüpfte Amazon DocumentDB-Rollen

Amazon DocumentDB unterstützt die Verwendung von serviceverknüpften Rollen in allen Regionen, in denen der Service verfügbar ist. Weitere Informationen finden Sie unter https://docs.aws.amazon.com/documentdb/latest/devguide/regions-and-azs.html#regions-and-azs-availability.

Leistungsoptimierung

Dieses Kapitel enthält Anleitungen zur Optimierung der Amazon DocumentDB-Leistung und zur Behebung häufiger Leistungsprobleme.

Themen

- Allgemeine Leistungsprobleme
- Anti-patterns
- Tipps zur Leistungsverbesserung
- Plananalyse abfragen
- Index-only Scans (abgedeckte Abfragen)
- Cache-Filter planen
- Bewährte Methoden für Amazon DocumentDB
- Amazon DocumentDB-Indizes verwalten
- Verwaltung der Dokumentenkomprimierung auf Sammlungsebene
- Verwaltung der wörterbuchbasierten Komprimierung in Amazon DocumentDB 8.0

Allgemeine Leistungsprobleme

In diesem Abschnitt werden allgemeine Leistungsprobleme behandelt, auf die Sie bei Amazon DocumentDB stoßen können, und wie Sie diese lösen können.

Themen

- Hohe CPU-Auslastung
- Verbindungsprobleme mit Amazon DocumentDB
- Abfrage läuft langsam

Hohe CPU-Auslastung

Eine hohe CPU-Auslastung ist eines der Probleme, auf die Sie bei der Arbeit mit Amazon DocumentDB irgendwann stoßen könnten. Dieser Abschnitt bietet Einblicke zur Lösung des Problems.

Identifizierung — Finden Sie das Problem

Eine hohe CPU-Auslastung auf Amazon DocumentDB DocumentDB-Instances führt zu einer Verlangsamung der Anwendungen, da die Latenzen bei allen Datenbankoperationen erhöht werden. Wenn die CPU-Auslastung von Amazon DocumentDB den optimalen Wert überschreitet, verlängern sich die Ausführungszeiten von Abfragen, der Verbindungsaufbau dauert länger und die Reaktionsfähigkeit der Anwendung nimmt ab. CPU-Spitzen auf der primären Instance verlangsamen Schreibvorgänge, während Spitzenwerte bei Replikaten zu einer höheren Latenz bei Lesevorgängen führen.

Einige der häufigsten Ursachen für eine hohe CPU-Auslastung in Amazon DocumentDB könnten sein:

- Sammlungsscans: Abfragen ohne korrekte Indizes erzwingen vollständige Sammelscans, wodurch übermäßig viel CPU zur Untersuchung jedes Dokuments verbraucht wird.
- Verbindungsspitzen: Ein plötzlicher Anstieg der Anwendungsverbindungen überfordert die für Authentifizierung und Sitzungsverwaltung benötigten CPU-Ressourcen.
- Komplexe Aggregationen: Multi-stage Operationen (Sortieren, Gruppieren, Zusammenfügen) verbrauchen viel CPU, insbesondere bei großen Datensätzen ohne optimierte Indizes.
- Ineffiziente Abfragen: Eine hohe CPU-Auslastung resultiert aus dem Scannen großer Ergebnismengen und Regex-Operationen.
- Write-Heavy Workloads: Häufige Aktualisierungen und Masseneinfügungen belasten die CPU bei der Dokumentenverarbeitung, Indexaktualisierungen und Transaktionsprotokollierung.

Untersuchen — Metriken sammeln

Amazon DocumentDB bietet Überwachung über Amazon CloudWatch. Amazon DocumentDB-Metriken können grob in Metriken auf Cluster- und Instance-Ebene unterteilt werden. Sehen Sie sich bei Ressourcen, die sich auf CPU, Arbeitsspeicher und Verbindungen beziehen, die Metriken auf Instance-Ebene an, da der Ressourcenverbrauch für die Instance spezifisch ist. Identifizieren Sie die Instances (primär oder repliziert), die CPU-Spitzen anzeigen, und notieren Sie sich die Zeiten der Spitzen.

Die CloudWatch Metrik für die CPU-Auslastung für. `CPUUtilization`

Diagnose — Finden Sie die Grundursache

Nachdem Sie die Instanz und den Zeitplan der CPU-Spitze identifiziert haben, müssen Sie als Nächstes die Ursache diagnostizieren, die die CPU-Auslastung erheblich beeinträchtigt. Da es mehrere Gründe gibt, die zu einem CPU-Anstieg führen können, wollen wir zunächst herausfinden, was dazu beiträgt:

Verbindungen: Überprüfen Sie unter anderem folgende CloudWatch `DatabaseConnections` Kennzahlen: Falls es Verbindungsspitzen gibt und diese mit der CPU-Spitzenzeit zusammenfallen, ist der Anstieg der Verbindungen über einen kurzen Zeitraum wahrscheinlich die Ursache für den CPU-Anstieg. `DatabaseConnectionsMax`

Abfragen: Wenn es nicht die Verbindung ist, könnte eine Abfrage die Ursache sein. Wenn Sie Performance Insights aktiviert haben, rufen Sie die Performance Insights-Konsole für die Instance auf und sehen Sie sich die Abfragen auf der Instance an. Die andere Sache, mit der Sie sich Abfragen ansehen können, stammt aus dem Profiler.

Lösen — Das Problem beheben

Verbindungsbursts: Wenn Verbindungsbursts die Ursache für CPU-Spitzen sind, sollten Sie erwägen, Verbindungen in Ihrer Anwendung wiederzuverwenden oder Verbindungspooling zu implementieren. Weitere Informationen zur Optimierung von Verbindungen mit hohem Verbindungsaufkommen finden Sie unter. the section called “Verbindungsprobleme”

Abfrageverteilung: Wenn die CPU-Spitzen durch Leseabfragen auf der primären Instanz verursacht werden, sollten Sie erwägen, Lesevorgänge auf Replikatinstanzen umzuleiten, wenn Ihre Abfrageergebnisse letztendlich konsistent sein können.

Abfrageoptimierungen: Wenn eine Abfrage oder eine Reihe von Abfragen zu einem CPU-Anstieg führen, sollten Sie in Erwägung ziehen, die Abfrage zu optimieren, z. B. einen Index hinzuzufügen, falls die Abfrage keinen verwendet. Weitere Informationen zur Optimierung einer Abfrage finden Sie unter. the section called “Die Abfrage läuft langsam”

Instanzskalierung: Wenn die CPU-Auslastung nach der Optimierung konstant hoch bleibt, sollten Sie auf eine größere Instance-Klasse mit mehr CPU-Kapazität aufrüsten, um Ihre Workload-Anforderungen effektiv zu bewältigen.

Serverlos: Bei unvorhersehbaren Workloads, die zu kurzzeitigen CPU-Spitzen führen, sollten Sie die Verwendung von. Serverlose Verwendung von Amazon DocumentDB Serverlose Instances skalieren

Ressourcen automatisch je nach Bedarf, sodass für solche Workloads keine festen Instanzgrößen bereitgestellt werden müssen.

Verbindungsprobleme mit Amazon DocumentDB

Identifizierung — Ermitteln Sie das Problem

Häufige Ursachen

Verbindungsprobleme sind in der Regel auf drei Hauptbereiche zurückzuführen:

Der Verbindungspool ist erschöpft, wenn eine Anwendung ihre maximal zulässige Anzahl an Verbindungen zu Amazon DocumentDB erreicht, entweder über die clientseitigen Grenzwerte für den Verbindungspool oder über die serverseitigen Instance-Grenzwerte. Dieser Zustand führt zu einer verminderten Anwendungsleistung, Timeouts und potenziellen Ausfällen, da neue Verbindungsanfragen entweder in die Warteschlange gestellt oder abgelehnt werden.

Eine Überlastung der Authentifizierung tritt auf, wenn Amazon DocumentDB zu viele gleichzeitige Authentifizierungsanfragen erhält, insbesondere bei der Verarbeitung von mehr als 1.000 neuen Verbindungen in einem kurzen Zeitraum. Während der Wartung der Authentifizierung hält Amazon DocumentDB eine exklusive Sperre auf der Sitzungsübersicht fest, sodass nachfolgende Authentifizierungsversuche in die Warteschlange geraten, bis die Wartung abgeschlossen ist.

Konfigurationsprobleme in Amazon DocumentDB sind häufig auf Fehlkonfigurationen in den Netzwerk-, Sicherheits- und Client-Einstellungen zurückzuführen. Dazu gehören Elemente wie falsche Sicherheitsgruppeneinstellungen, falsche VPC-Konfiguration oder SSL/TLS Zertifikatsprobleme. Das Verständnis der richtigen Konfiguration ist für einen sicheren und zuverlässigen Datenbankzugriff unerlässlich.

Diagnose — Finden Sie die Grundursache

Verbindungspools

Der Verbindungspool wird beim Erstellen einer MongoClient Instanz initialisiert. Jeder Pool unterhält Verbindungen auf der Grundlage von zwei Schlüsselparametern:

min PoolSize — Mindestanzahl der verwalteten Verbindungen

max PoolSize — Maximal zulässige Verbindungen

Wenn eine Anfrage eine Verbindung benötigt:

1. Der Pool sucht nach verfügbaren Verbindungen im Leerlauf
2. Wenn keine vorhanden ist und die Poolgröße < maximal istPoolSize, wird eine neue Verbindung erstellt
3. Wenn der Maximalwert erreicht PoolSize ist, wird die Anfrage in eine Warteschlange gestellt
4. Wenn die Warteschlange voll ist oder das Timeout erreicht ist, wird Folgendes ausgelöst MongoWaitQueueFullException

Das Verhalten der Wartewarteschlange wird über diese Parameter gesteuert:

wait QueueTimeout MS — Maximale Wartezeit für die Verbindung

wait QueueSize — Maximale Anzahl von Anfragen in der Warteschlange

Hier ist ein Beispiel für einen problematischen Ansatz für die Verbindung zu Amazon DocumentDB, bei dem jedes Mal ein neuer Pool erstellt wird:

```
for(Request request : requests) {
    MongoClient client = MongoClients.create(settings);
    // Process request
    client.close();
}
```

Zu den wichtigsten zu überwachenden CloudWatch Kennzahlen gehören:

- `DatabaseConnections`— Die Anzahl der Verbindungen (aktiv und inaktiv), die auf einer Instance geöffnet werden, und zwar im Abstand von 1 Minute.
- `DatabaseConnectionsMax`— Die maximale Anzahl offener Datenbankverbindungen (aktiv und inaktiv) auf einer Instance in einem Zeitraum von 1 Minute.
- `DatabaseConnectionsLimit`— Die maximale Anzahl gleichzeitiger Datenbankverbindungen (aktiv und inaktiv), die auf einer Instance zu einem bestimmten Zeitpunkt zulässig sind.
- `LowMemNumOperationsThrottled`- Die Anzahl der Anfragen, die in einem Zeitraum von 1 Minute aufgrund von zu wenig verfügbarem Speicher gedrosselt werden.

Informationen zu Kontingenten pro Instance-Klasse finden Sie unter. Kontingente für Instanzen

Zu den häufigsten Warnsignalen für Probleme mit dem Connection Pool auf Anwendungsebene gehören:

- Zunehmende Zeiten für den Verbindungsaufbau
- Zunehmende Größe der Warteschlange
- Steigende Anzahl von Timeout-Ausnahmen

Überlastung der Authentifizierung

Das Herstellen einer Verbindung zu Amazon DocumentDB folgt einem ähnlichen Ablauf wie im Folgenden:

Verbindungsanfrage → SSL-Handshake → Authentifizierung → Sitzungserstellung → Verbindung bereit

Wenn mehr als 1.000 neue Verbindungen verarbeitet werden, werden zusätzliche Verbindungsanfragen nach Abschluss des SSL-Handshakes in eine Warteschlange für die Authentifizierung gestellt. Während dieser Überlastungsereignisse erhöht sich die durchschnittliche Verbindungsdauer Ihrer Anwendung.

Zu den CloudWatch wichtigsten zu überwachenden Kennzahlen gehören:

- `DatabaseConnections`— Die Anzahl der Verbindungen (aktiv und inaktiv), die auf einer Instance geöffnet werden, und zwar im Abstand von 1 Minute.
- `DatabaseConnectionsMax`— Die maximale Anzahl offener Datenbankverbindungen (aktiv und inaktiv) auf einer Instance in einem Zeitraum von 1 Minute.
- `DatabaseConnectionsLimit`— Die maximale Anzahl gleichzeitiger Datenbankverbindungen (aktiv und inaktiv), die auf einer Instance zu einem bestimmten Zeitpunkt zulässig sind.

- Probleme mit der Konfiguration

Das häufigste Konfigurationsproblem wird verursacht, wenn versucht wird, von einer Umgebung ohne Zugriff auf die private Netzwerkumgebung aus eine Verbindung zu einem privaten Amazon DocumentDB-Cluster-Endpunkt herzustellen. Amazon DocumentDB ist nur für Virtual Private Cloud (VPC) verfügbar und unterstützt derzeit keine öffentlichen Endpunkte. Sie können von Ihrem Laptop oder Ihrer lokalen Entwicklungsumgebung außerhalb Ihrer VPC aus keine direkte Verbindung zu Ihrem Amazon DocumentDB-Cluster herstellen.

Dies wird sich in Fehlern wie den folgenden äußern:

```
Error: couldn't connect to server...
Failed to connect to...
exception: connect failed
connection attempt failed
```

Falsche Sicherheitsgruppenkonfigurationen können auch zu Verbindungsfehlern führen. Ein Amazon DocumentDB-Cluster wartet standardmäßig auf Verbindungen auf dem TCP-Port 27017. Ihre Anwendung schlägt fehl, wenn Sie versuchen, eine Verbindung zu einem anderen Port herzustellen als dem, mit dem der Cluster bereitgestellt wurde, oder wenn die Anwendung nicht in der Konfiguration der Ingress-Sicherheitsgruppe für den Cluster enthalten ist.

Eine falsche Zertifikatsverwaltung kann auch zu Verbindungsproblemen führen. Standardmäßig ist die Verschlüsselung bei der Übertragung für neu erstellte Amazon DocumentDB-Cluster aktiviert. Wenn die Verschlüsselung während der Übertragung aktiviert ist, sind sichere Verbindungen mit TLS erforderlich, um mithilfe des global-bundle.pem-Zertifikats eine Verbindung zum Cluster herzustellen. Wenn Sie versuchen, das falsche Zertifikat zu verwenden, erhalten Sie Fehler wie:

```
unable to get local issuer certificate
```

Wenn Sie versuchen, eine Verbindung zu einem Cluster mit aktiviertem TLS herzustellen, ohne die TLS-Parameter anzugeben, erhalten Sie Fehler wie:

```
Server selection timed out after 30000 ms
```

Gelöst — Beheben Sie das Problem

Verbindungspools: Überprüfen Sie das Verbindungspooling, indem Sie die Poolgrößen implementieren oder an die Workload-Anforderungen anpassen. Optimale Poolkonfigurationen hängen von Ihrer Arbeitslast und Ihren Anforderungen ab. Sie sollten ein Minimum PoolSize einhalten, damit die Kernverbindungen bereit und verfügbar sind, und ein Maximum, das WaitTime kurz genug ist, um schnell auszufallen, wenn der Pool erschöpft ist.

Hier ist ein Beispiel dafür, wie Sie einen einzelnen Pool wiederverwenden können, ohne jedes Mal einen neuen zu erstellen:

```
MongoClient client = MongoClients.create(settings);
```

```
    for(Request request : requests) {
    // Process request
}
```

Überlastung durch Authentifizierung: Verwalten Sie die Authentifizierung, indem Sie einen schrittweisen Verbindungsaufbau implementieren und die Anzahl neuer Verbindungen auf jeweils 1.000 beschränken. Verwenden Sie Verbindungspooling, um authentifizierte Verbindungen effektiv wiederzuverwenden. Um zu vermeiden, dass der Amazon DocumentDB-Cluster mit Verbindungen überlastet wird, implementieren Sie eine Strategie zum Verbindungsaufbau.

```
public class ConnectionManager {
    private static final int BATCH_SIZE = 100;
    private static final int DELAY_MS = 1000;

    public void establishConnections(int totalRequired) {
        int established = 0;
        while (established < totalRequired) {
            int batch = Math.min(BATCH_SIZE, totalRequired - established);
            createConnections(batch);
            Thread.sleep(DELAY_MS);
            established += batch;
        }
    }
}
```

Sie können Ihre Verbindungspool-Einstellungen auch so konfigurieren, dass die Gesamtzahl der zulässigen Verbindungen begrenzt wird.

```
MongoClientSettings settings = MongoClientSettings.builder()
    .applyToConnectionPoolSettings(builder -> {
        builder.maxSize(500)                    // Limit total connections
               .minSize(10)                     // Maintain base connections
               .maxConnectionLifeTime(3600000)  // Rotate connections hourly
    })
    .applyToServerSettings(builder -> {
        builder.heartbeatFrequency(10000)       // Regular server checks
    })
    .build();
```

Konfigurationsprobleme: Stellen Sie sicher, dass Ihre Anwendung Zugriff auf die private VPC und das Subnetz hat, in denen sich Ihre Amazon DocumentDB-Ressourcen befinden. Wenn Sie VPC-Peering verwenden, finden Sie weitere Informationen im Entwicklerhandbuch Problembehandlung bei einer VPC-Peering-Verbindung. Sie können auch den Knowledge Center-Artikel Wie behebe ich Verbindungsprobleme zwischen dem Internet und Amazon EC2-Instances in meiner VPC? .

Für die Konfiguration der Sicherheitsgruppe müssen Sie eine Eingangsregel in Ihre Amazon DocumentDB-Sicherheitsgruppe aufnehmen, um Verbindungen von Ihrer Anwendung aus zuzulassen.

```
{
  "SecurityGroupIngress": [
    {
      "IpProtocol": "tcp",
      "FromPort": 27017,
      "ToPort": 27017,
      "SourceSecurityGroupId": "<application-security-group>",
      "Description": "DocumentDB access from application tier"
    }
  ],
  "SecurityGroupEgress": [
    {
      "IpProtocol": "-1",
      "FromPort": -1,
      "ToPort": -1,
      "CidrIp": "0.0.0.0/0"
    }
  ]
}
```

Wenn der Cluster mit TLS-Verschlüsselung konfiguriert ist, laden Sie das TLS-Zertifikat für Amazon Amazon DocumentDB mit dem Namen global-bundle.pem herunter und verwenden Sie es, wenn Sie eine Verbindung zum Cluster herstellen.

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

Long-term Lösungen

Eine Skalierung der Instance kann durch ein Upgrade auf eine größere Instance-Klasse oder das Hinzufügen von Read Replicas zur Verteilung der Verbindungslast erforderlich sein. Die korrekte Implementierung des Lastenausgleichs gewährleistet eine optimale Nutzung der Ressourcen im gesamten Cluster.

Anwendungsänderungen sollten sich auf die Implementierung einer robusten Verbindungsverwaltung, eine umfassende Überwachung und die Einhaltung der bewährten Methoden für das Verbindungspooling konzentrieren. Dazu gehören die richtige Fehlerbehandlung und das Management des Verbindungslebenszyklus.

Zu den Verbesserungen der Architektur gehören möglicherweise die Einführung von Amazon DocumentDB Serverless für variable Workloads, die Implementierung einer ausgeklügelten Wiederholungslogik und das Design im Hinblick auf Fehlertoleranz. Erwägen Sie, die Anwendungsarchitektur umzustrukturieren, um das Verbindungsmanagement besser zu handhaben.

Bewährte Methoden

Verbindungspools

Durch die richtige Verwaltung und Überwachung des Verbindungspools können Anwendungen eine stabile Datenbankkonnektivität aufrechterhalten und gleichzeitig Überlastungsszenarien verhindern, die sich auf die Zuverlässigkeit und Leistung des Systems auswirken könnten. Konfigurieren Sie angemessene Timeouts und passen Sie Ihren Pool entsprechend den Merkmalen Ihrer Arbeitslast an.

Beispiel für die Einstellung eines Verbindungspools

```
MongoClientSettings settings = MongoClientSettings.builder()
    .applyToConnectionPoolSettings(builder ->
        builder.maxSize(10))
    .applyToConnectionPoolSettings(builder ->
        builder.maxWaitQueueSize(2))
    .applyToConnectionPoolSettings(builder ->
        builder.maxConnectionIdleTime(10, TimeUnit.MINUTES))
    .build();
```

Weitere Informationen finden Sie unter Erstellen robuster Anwendungen mit Amazon DocumentDB — Teil 1: Client-Konfiguration.

Überlastung bei der Authentifizierung

Implementieren Sie das Verbindungspooling immer mit entsprechenden Werten für Parameter, die auf Ihrer Arbeitslast basieren. Verwenden Sie ein schrittweises Verfahren zum Verbindungsaufbau und pflegen Sie, wenn möglich, persistente Verbindungen. Implementieren Sie eine ordnungsgemäße Verbindungsbereinigung, um sicherzustellen, dass keine ungenutzten Ressourcen verschwendet werden.

Probleme mit der Konfiguration

Stellen Sie sicher, dass Sie das richtige Routing von Ihrer Anwendung zu den Amazon DocumentDB-Ressourcen konfiguriert haben. Verwenden Sie TLS für die Verschlüsselung während der Übertragung und implementieren Sie den Zugriff mit den geringsten Rechten. Überprüfen Sie Ihre Amazon DocumentDB-Anmeldeinformationen und überprüfen Sie die Werte der Verbindungszeichenfolge.

Abfrage läuft langsam

Identifizierung — Finden Sie das Problem

Eine langsam laufende Abfrage ist eine, die die für Ihre Geschäftsanforderungen erwartete Ausführungszeit überschreitet. Die Definition, was eine langsame Abfrage ausmacht, ist je nach Workload unterschiedlich. Sie können langsame Abfragen anhand von Profiler-Protokollen oder Performance Insights identifizieren.

1. Aktivieren Sie den Profiler, falls er nicht bereits aktiviert ist. Profiler-Protokolle werden CloudWatch unter der Protokollgruppe: `/aws/docdb/<cluster-identifier>/profiler` auf Amazon geschrieben. Verwenden Sie CloudWatch Logs Insights, um sie abzufragen.

   Beispiel für eine CloudWatch Log-Analyse-Abfrage, um die 10 langsamsten Abfragen für die Sammlung ecommerce.products abzurufen:

   ```
   filter ns="ecommerce.products" | sort millis desc | limit 10
   ```

2. Verwenden Sie Performance Insights, um teure Abfragen nahezu in Echtzeit zu identifizieren. Aktivieren Sie Performance Insights, falls es noch nicht aktiviert ist.
   a. Öffnen Sie die AWS Konsole, navigieren Sie zu Amazon Amazon DocumentDB, dann zu Performance Insights und wählen Sie Ihre Cluster-Instance aus.

b. Überprüfen Sie den Zeitplan für DB-Ladevorgänge (AAS) und die häufigsten Abfragen (nach DB-Auslastung). Erweitern Sie einen Abfrage-Digest, um wörtliche untergeordnete Anweisungen anzuzeigen.

c. Erfassen Sie die Abfragen, die analysiert werden müssen.

> **Note**
>
> Möglicherweise sind nicht alle Abfragen in Performance Insights ineffiziente oder langsame Abfragen.

Untersuchen — Informationen sammeln

1. Der Profiler stellt den Ausführungsplan für die Abfrage und die damit verbundenen wichtigsten Kennzahlen bereit, darunter millis, nreturned und PlanSummary (Indexnutzung):


```
{
    "op": "query",
    "ts": 1721374275673,
    "ns": "test.perf",
    "command": {
        "find": "perf",
        "filter": {
            "threadRunCount": 0
        },
        "$db": "test",
        "lsid": {
            "id": {
                "$binary": "oO2wEtpgQIK+y9KGByYnsw==",
                "$type": "4"
            }
        },
        "$readPreference": {
            "mode": "secondaryPreferred"
        }
    },
    "cursorExhausted": true,
    "nreturned": 0,
    "responseLength": 0,
    "protocol": "op_query",
```

```
    "millis": 137,
    "planSummary": "IXSCAN",
    "execStats": {
        "stage": "FETCH",
        "nReturned": "0",
        "executionTimeMillisEstimate": "100.346",
        "inputStage": {
            "stage": "IXSCAN",
            "nReturned": "0",
            "executionTimeMillisEstimate": "100.342",
            "indexName": "threadRunCount_1"
        }
    },
    "client": "172.31.6.165:43154",
    "appName": "ProdAppTester14",
    "user": "adminuser"
}
```

Um die COLLSCAN-Abfragen zu finden:

```
filter planSummary="COLLSCAN" | sort millis desc | limit 20
```

2. Verwenden Sie Performance Insights, um Trends bei der Abfrageausführung wie Wartezeiten, Auslastung und Ressourcenauswirkungen (z. B. IOPS oder CPU pro Abfrage) in Echtzeit zu analysieren.

   Da Performance Insights den Ausführungsplan für die Abfrage nicht bereitstellt, erfassen Sie die Abfrage und führen Sie explain („executionStats“) für die Abfrage in Ihrem Amazon DocumentDB-Cluster aus:

   ```
   db.ecommerce.products.find().explain("executionStats")
   ```

3. Korrelieren Sie optional Profiler-Metriken mit Performance Insights-Daten (ordnen Sie z. B. hohe Millionenabfragen im Profiler den häufigsten Abfragen in Performance Insights zu).

Diagnose — Finden Sie die Ursache

In diesem Schritt diagnostizieren Sie den Abfrageplan, um mögliche Optimierungen zu identifizieren. Verwenden Sie das folgende Flow-Symptom → wahrscheinliche Ursache:

- Symptom: PlanSummary: „COLLSCAN“

  Ursache: Fehlender oder falscher Index.
- Symptom: Langsame Aggregation mit $group/$sort

  Ursache: Die Aggregationspipeline verarbeitet möglicherweise zu viele Daten im Speicher.
- Symptom: hohe Latenzen, während Performance Insights DB Load gruppiert nach I/O-Wartezeiten anzeigt, obwohl Indizes verwendet werden (PlanSummary: „IXSCAN“).

  Ursache: I/O gebundene Abfragen; Indizes oder Arbeitssatz überschreiten den verfügbaren Puffercache auf der Instance.
- Symptom: PI zeigt CPU-Wartestatus und hohe AAS aufgrund weniger Abfragen

  Ursache: CPU-bound Abfragen (komplexe Aggregationen, $regex).
- Symptom: Viele langsame Abfragen, aber keine zeigt teure PlanSummary

  Ursache: externer Engpass (Netzwerk, Drosselung, Wartungsaufgaben, Anzahl der Abfragen).

Beheben — Das Problem beheben

Wenn Sie Fixes implementieren, validieren Sie immer mit explain („executionStats“) und überwachen Sie Performance Insights DB Load.

1. Zusammenfassung des Plans: „COLL SCAN“
   - Erstellen Sie einen zielgerichteten Index.

     Beispiel: Für häufige Abfragen, die nach {Kategorie, Preis} filtern und nach absteigendem Preis sortieren:

     ```
     db.products.createIndex({ category: 1, price: -1 })
     ```
2. langsame Aggregation mit $group/$sort

- Setzen Sie $match und $project frühzeitig ein, um zu verhindern, dass Dokumente in $group/$sort landen.
- Begrenzen Sie die Anzahl der Felder zu Beginn der Pipeline:

```
db.products.explain("executionStats").aggregate([
  { $match: { category: "Electronics", price: { $gt: 0 } } },   // early filter
  { $project: { price: 1, category: 1 } },                     // reduce document
 size
  { $group: { _id: "$category", avgPrice: { $avg: "$price" } } }
])
```

3. hohe Latenzen, während Performance Insights die DB-Auslastung gruppiert nach I/O-Wartezeiten anzeigt
   - Überprüfen Sie, ob BufferCacheHitRatio der Wert niedrig ist oder ob die ReadIOPS auf der Instance hoch sind. Erhöhen Sie den Instanzspeicher (aktualisieren Sie die Instanzklasse, z. B. r6g.large → r6g.xlarge) oder verwenden Sie die NVMe-Instanzklasse.
   - Reduzieren Sie den Platzbedarf im Index.
   - Fügen Sie Read Replicas hinzu, um den Leseverkehr auszulagern (verwenden Sie die ReadPreference-Einstellung, um den Datenverkehr auf Replikaten umzuleiten, wenn die Abfrage tolerant und letztendlich konsistent ist).
4. PI zeigt CPU-Wartezustände an, hohe AAS aufgrund weniger Abfragen
   - Ersetzen Sie teure $regex durch indizierte Präfixsuchen oder $text-Index.
   - Batch-Schreibvorgänge (Einfügen), um die Schreibverstärkung zu reduzieren.

> **Note**
>
> Testen Sie Ihre Änderungen immer in einer niedrigeren Umgebung (außerhalb der Produktionsumgebung), bevor Sie diese Änderungen in die Produktionsumgebung übertragen.

Anti-patterns

In diesem Abschnitt werden gängige Anti-Pattern-Methoden behandelt, die sich negativ auf die Leistung von Amazon DocumentDB auswirken können.

Themen

- Zusammengesetzter Index mit mehr als 3 Attributen
- Abfragen mit langer Laufzeit
- Wenig genutzte und redundante Indizes
- Multi-Key Indizes mit großen Arrays

Zusammengesetzter Index mit mehr als 3 Attributen

-Übersicht

Ein zusammengesetzter oder zusammengesetzter Index verwaltet Verweise auf mehrere Felder innerhalb einer einzigen Indexstruktur. Diese Indizes optimieren die Leistung für Abfragen, die nach mehreren Feldern gleichzeitig filtern oder Filterung mit Sortiervorgängen kombinieren. Sie eignen sich auch für Abfragen mit einer einzigen Bedingung in den indizierten Feldern ganz links. Die Datenbank nutzt diese Indexeinträge, um passende Dokumente effizient zu finden, ohne die Sammlung vollständig scannen zu müssen.

Amazon DocumentDB kann den zusammengesetzten Index verwenden, um Abfragen zu unterstützen, die eine beliebige führende Teilmenge der indizierten Felder enthalten. Dieses Konzept wird als Indexpräfixe bezeichnet. Wenn Sie beispielsweise einen zusammengesetzten Index aktiviert haben`{state: 1, city: 1, zipcode: 1}`, kann Amazon DocumentDB Abfragen effizient verarbeiten, die nur das Feld „state“, die Felder „state“ und „city“ zusammen oder alle drei Felder „state“, „city“ und „zip code“ verwenden. Abfragen müssen jedoch die Felder von links nach rechts verwenden, ohne dass Felder dazwischen übersprungen werden. Das bedeutet, dass Abfragen, die nur das Feld „Stadt“ oder „Postleitzahl“ oder Kombinationen wie „Bundesland“ und „Postleitzahl“ (ohne „Stadt“) verwenden, den Index nicht vollständig nutzen können.

In den meisten realen Szenarien erzielen zusammengesetzte Indizes mit drei oder weniger Attributen in der Regel optimale Leistung und Ressourceneffizienz. Es ist zwar möglich, mehr als 3 Attribute im zusammengesetzten Index zu haben, führt jedoch häufig zu einem erhöhten Ressourcenverbrauch, der die Vorteile überwiegt.

Auswirkungen auf den Cluster

Es ist zwar verlockend, einen Index zu erstellen, der alle Bedingungen in einer Abfrage abdeckt, um eine maximale theoretische Leistung zu erzielen, aber die meisten Datenfilterung erfolgt über die ersten 1—3 Attribute in einem zusammengesetzten Index. Zusätzliche Felder, die diesen Schwellenwert überschreiten, tragen in erster Linie zur Indexgröße und nicht zur sinnvollen Abfrageoptimierung bei.

Speicherplatz und I/O Overhead: Zusammengesetzte Indizes mit vielen Attributen verbrauchen deutlich mehr Speicherplatz als einfachere Alternativen. Die Größe ist direkt proportional zur Anzahl der indizierten Attribute und zur Größe der indizierten Werte selbst.

Speicherbedarf: Der große Speicherbedarf zusammengesetzter Indizes mit vielen Attributen führt zu einem entsprechenden Speicherbedarf, wodurch Ihr Arbeitssatz größer wird und andere häufig aufgerufene Daten aus dem Pufferpool verdrängt werden.

Schreiboperationen: Jede Änderung eines Dokuments, die sich auf mehrere indizierte Felder auswirkt, erfordert die Aktualisierung des gesamten zusammengesetzten Indexeintrags, wodurch der Arbeitsaufwand für den Schreibvorgang vervielfacht wird.

Wie identifiziert man

Überprüfen Sie zunächst alle Indizes in Ihrer Sammlung, um zusammengesetzte Indizes mit mehr als drei Attributen zu identifizieren:

```
// List all indexes for the collection
db.collection.getIndexes()

// Look for indexes with 3+ fields like:

// { "userId": 1, "status": 1, "category": 1, "priority": 1, "region": 1 }
// { "orderId": 1, "customerId": 1, "productId": 1, "timestamp": 1, "warehouse": 1 }
```

Abhilfe

Wenn ein zusammengesetzter Index mit mehr als drei Attributen verwendet wird, identifizieren Sie die Abfragen, die ihn verwenden, und suchen Sie nach Optimierungsmöglichkeiten. Erwägen Sie, diese Indizes durch effizientere Indizes mit drei oder weniger Attributen zu ersetzen. Entfernen Sie nach der Implementierung der neuen Indizes die alten Indizes, die mehr als drei Attribute enthalten.

> **Note**
>
> Stimmen Sie sich stets mit den Stakeholdern ab und überprüfen Sie die Auswirkungen auf die Leistung, bevor Sie Indizes löschen.

Halten Sie sich bei der Erstellung zusammengesetzter Indizes an die ESR-Regel (Equality, Sort, Range).

Equality Sort Range (ESR) -Regel: Diese Reihenfolge maximiert die Indexeffizienz, indem der Datensatz zunächst nach Gleichheitsbedingungen gefiltert wird, dann Sortieroperationen auf die reduzierte Menge angewendet und schließlich Bereichsscans für die kleinstmögliche Teilmenge durchgeführt werden

- Felder nach Gleichheit (exakte Treffer) anordnen,
- Sortieren (Reihenfolge),
- Bereich (>, <, $in).

```
// Query pattern
db.orders.find({
  userId: "user123",                    // Equality
  price: { $gte: 50, $lte: 200 }      // Range
}).sort({ createdAt: -1 })             // Sort

// Optimal index following ESR rule
db.orders.createIndex({ userId: 1, createdAt: -1, price: 1 })
//                      Equality   Sort         Range
```

Abfragen mit langer Laufzeit

-Übersicht

Long-running Abfragen können zu clusterweiten Leistungsproblemen führen. Diese Abfragen können den Garbage-Collection-Prozess von Amazon DocumentDB (Multi-Version Concurrency Control, MVCC) stören, was zur Anhäufung von Dokumenten mit alten Versionen und zu Leistungseinbußen im gesamten Cluster führen kann. Amazon DocumentDB implementiert ein 2-stündiges serverseitiges

Timeout als Sicherheitsmechanismus, um zu verhindern, dass außer Kontrolle geratene Abfragen auf unbestimmte Zeit Ressourcen verbrauchen.

Was sind Abfragen mit langer Laufzeit:

- Abfragen, die über einen längeren Zeitraum (in der Regel > 30 Minuten) ausgeführt werden
- Öffnen Sie Cursor, die stundenlang aktiv bleiben (ein Timeout von 2 Stunden gilt nicht, wenn der Cursor aktiv ist)

Auswirkungen auf den Cluster

Abfragen mit langer Laufzeit können den Garbage-Collection-Prozess beeinträchtigen

- Alte Versionen häufen sich an: Garbage Collector kann alte Dokumentversionen nicht zurückfordern
- Aufblähung von Sammlung und Index: Sammlungs- und Indexeinträge häufen sich im Laufe der Zeit an, die Aufblähung nimmt zu, was zu höheren Speicherkosten führen kann.
- CPU- und Speicherauslastung: Die CPU- und Speicherauslastung steigt aufgrund der ineffizienten Verarbeitung einer erhöhten Anzahl alter Dokumentversionen, Indexeinträge und Transaktions-IDs.

Abfrage mit langer Laufzeit → Blocks-GC → Speicherwachstum → CPU- und Speicherauslastung → Mehr lange Abfragen

Überwachen und erkennen

1. Verwenden Sie den Befehl currentOp, um Abfragen mit langer Laufzeit zu finden.

```
// To find a query running for more than 30 mins
db.adminCommand({
    aggregate: 1,
    pipeline: [
        {$currentOp: {}},
        {$match:
            {$or:
                [{secs_running: {$gt: 1800}},
                 {WaitState: {$exists: true}}]}}],
    cursor: {}
});
```

2. Um Cursor zu finden, die länger als 30 Minuten aktiv sind

```
// To find cursor which is running more than 30 mins
db.adminCommand({
    "currentOp": true,
    "active": true,
    "$all": true
}).inprog.filter(function(op) {
    return op.desc == "Cursor" &&
           op.secs_running > 1800 &&
           op.active == true;
}).sort((a, b) => b.microsecs_running - a.microsecs_running)
```

3. Überwachung des Fortschritts des Garbage Collectors durch CloudWatch

`LongestRunningGCProcess`— Dauer des längsten aktiven Garbage-Collection-Vorgangs in Sekunden. Wird jede Minute aktualisiert und verfolgt nur aktive Vorgänge, ausgenommen Prozesse, die innerhalb des einminütigen Zeitfensters abgeschlossen werden.

`AvailableMVCCIds`-Ein Zähler, der die Anzahl der verbleibenden Schreiboperationen anzeigt, bevor der Wert Null erreicht wird. Wenn dieser Zähler Null erreicht, wechselt Ihr Cluster in den schreibgeschützten Modus, bis IDs zurückgewonnen und recycelt werden. Der Zähler nimmt mit jedem Schreibvorgang ab und steigt, wenn bei der Garbage-Collection alte MVCC-IDs wiederverwendet werden.

> **Note**
>
> Niedrigere MVCC-IDs und eine längere Garbage-Collection-Dauer sind nicht ausschließlich auf Abfragen mit langer Laufzeit zurückzuführen. Write-intensive Workloads auf Instanzen mit eingeschränkten Ressourcen können auch zu einer verringerten MVCC-ID-Verfügbarkeit und längeren Garbage-Collection-Zyklen führen.

Strategien zur Problembehebung

- Implementieren Sie Abfrage-Timeouts in der Anwendung

- Halten Sie die Cursor nicht für längere Zeit am Leben
- Optimieren Sie die Abfragen für eine bessere Leistung.
- Bevorzugen Sie das Stapeln von Schreibvorgängen

Wenig genutzte und redundante Indizes

-Übersicht

Indizes verbessern die allgemeine Abfrageleistung von Workloads, sind jedoch mit Kosten verbunden. Jeder Index verbraucht Speicherplatz und I/O Ressourcen, und Schreibvorgänge müssen alle Indizes in der Sammlung aktualisieren. Wenn Ihre Datenbank ungenutzte, unzureichend genutzte oder redundante Indizes enthält, führt dies zu einem Overhead für den Cluster, was zu Leistungseinbußen führt. Durch mehrere unnötige Indizes entsteht ein Anti-Pattern, das zu Leistungsproblemen, erhöhten Speicherkosten und höherem Betriebsaufwand führt.

Suboptimale Indizierungsszenarien

- Unbenutzte Indizes: Indizes, die für einmalige Abfragen oder in früheren Produktiterationen erstellt wurden und auf die mit aktuellen Abfragemustern nicht mehr zugegriffen wird. Um ungenutzte Indizes für eine Sammlung zu identifizieren, können Sie IndexStats verwenden. Weitere Informationen finden Sie unter the section called “Analysieren Sie die Indexnutzung und identifizieren Sie ungenutzte Indizes”.
- Redundante Indizes: Mehrere Indizes, die sich überschneidende Schlüsselmuster oder identische Abfragemuster abdecken. Beispiel: Ein Einzelschlüsselindex für Attribut A wird überflüssig, wenn ein zusammengesetzter Index sowohl das Attribut A als auch das Attribut B abdeckt, da der zusammengesetzte Index Abfragen für Attribut A allein verarbeiten kann.
- Over-Indexing: Erstellung von Indizes „für alle Fälle“, ohne die tatsächlichen Abfragemuster oder Leistungsanforderungen zu analysieren.
- Indizes mit niedriger Kardinalität: Indizes für Felder mit wenigen eindeutigen Werten (z. B. boolesche Felder, Statusflags), die nur einen minimalen Nutzen bei der Abfrageoptimierung bieten.

Auswirkungen auf die Cluster-Leistung

- Speicher und I/O: Jeder Index verbraucht Speicherplatz und I/O-Ressourcen und trägt so zu den Gesamtkosten bei. Wenn ein Index ungenutzt oder redundant ist, bedeutet dies unnötige Kosten für Ihren Cluster.

- Eingeschränkte Schreibleistung: Bei Einfüge-, Aktualisierungs- und Löschvorgängen müssen Indizes beibehalten werden, was zu einem Mehraufwand führt, der Ressourcen verbraucht. Ungenutzte oder redundante Indizes führen zu unnötigem Overhead bei diesen Vorgängen.
- Arbeitsspeicher- und CPU-Auslastung: Indizes konkurrieren um den Pufferpool-Speicher und können häufig aufgerufene Daten löschen, wenn der Arbeitssatz nicht in den Instanzspeicher passt, was zu einem suboptimalen Ergebnis führt. BufferCacheHitRatio Das System verbraucht außerdem CPU-Ressourcen, um die Indexstrukturen aufrechtzuerhalten.

Verfügbare Tools zur Identifizierung von Möglichkeiten zur Indexoptimierung

- Tool zur Indexüberprüfung:
- Tool zur Erkennung der Indexkardinalität

Zusammenstellung des Toolkits:

```
git clone https://github.com/awslabs/amazon-documentdb-tools.git
```

1. Tool zur Indexüberprüfung

Das Tool zur Indexüberprüfung analysiert alle Sammlungen und Indizes, um eine Liste der Indizes und ihrer Verwendungsmuster bereitzustellen, und identifiziert redundante Indizes in Ihrem Amazon DocumentDB-Cluster. Führen Sie dieses Tool für eine umfassende Analyse auf allen Cluster-Instances aus. Weitere Informationen finden Sie unter Index Review Tool.

Verwendung:

```
cd performance/index-review/

python3 index-review.py --server-alias <server-alias> --uri <mongodb-uri>
```

Ausgabedateien

- `{server-alias}-collections.csv`: Collection-level Statistiken
- `{server-alias}-indexes.csv`: Detaillierte Kennzahlen zur Indexnutzung
- `{server-alias}-{timestamp}-index-review.json`: Rohdaten für weitere Analysen

> **Note**
>
> Führen Sie es für jede Instanz im Cluster mit einem unabhängigen Alias aus, um Statistiken für jede Instanz zu generieren, und führen Sie dann das Tool aus, um eine kumulative Analyse zu erhalten

ZB: `python3 index-review.py --files "{server-alias}-{timestamp}-index-review.json, {server-alias}-{timestamp}-index-review.json" --server-alias full-review`

2. Tool zur Erkennung der Indexkardinalität

Dieses Tool identifiziert Indizes mit niedriger Kardinalität, die für die Abfrageleistung ineffizient sind.

Usage

```
cd performance/index-cardinality-detection/

python3 detect-cardinality.py --uri <mongodb-uri>
```

Parameter

- `--threshold`: Prozentsatz des Kardinalitätsschwellenwerts (Standard: 1%)
- `--sample-count`: Anzahl der zu prüfenden Dokumente (Standard: 100.000)
- `--max-collections`: Maximale Anzahl von Sammlungen, die pro Datenbank gescannt werden sollen (Standard: 100)

Weitere Informationen finden Sie unter https://aws.amazon.com/blogs/database/detect-and-fix-low-cardinality-indexes-in-amazon-documentdb/.

Strategien zur Problembehebung

1. Entfernung ungenutzter und redundanter Indizes

```
// Example: Drop unused index
```

```
db.collection.dropIndex("unused_index_name")
```

> **Note**
>
> Indizes sollten niemals gelöscht werden, ohne dies mit allen interessierten Parteien besprochen und die Leistung getestet zu haben.

2. Optimierung des Index mit niedriger Kardinalität

- Verwenden Sie Teilindizes mit Filtern
- Wandeln Sie einzelne Indizes mit niedriger Kardinalität in zusammengesetzte Indizes um

Multi-Key Indizes mit großen Arrays

-Übersicht

Multi-key Mit Indizes in Amazon DocumentDB können Sie Array-Felder effizient abfragen. Diese Indexierungsstrategie sorgt für deutlich geringere Abfragelatenzen bei der Suche innerhalb von Array-Daten und ist somit eine attraktive Option für Anwendungen mit komplexen Datenstrukturen.

Bei der Arbeit mit großen Arrays oder mehreren Array-Feldern innerhalb einer Sammlung ist es jedoch wichtig, die Auswirkungen auf die Ressourcen und die Leistungsmerkmale zu verstehen, um fundierte Architekturentscheidungen treffen zu können. Wenn Sie einen Index für ein Array-Feld erstellen, generiert Amazon DocumentDB individuelle Indexeinträge für jedes Element im Array.

Auswirkungen auf den Cluster

- Speicher- und I/O-Overhead: Multi-key Indizes beanspruchen manchmal Speicherplatz bis zu einem Vielfachen der Basistabelle. Die Größe ist direkt proportional zu den Dokumenten in der Sammlung mit dem indizierten Array-Attribut, der Anzahl der Elemente im Array und der Größe des Elements.
- Speicherverbrauch: Da der Speicherbedarf relativ groß ist, haben diese Indizes einen entsprechenden Speicherbedarf, was zu einem größeren Arbeitssatz führt.
- Schreiboperationen: Multi-key Indizes verursachen während der Schreibvorgänge Mehraufwand. Jedes Array-Element generiert separate Indexeinträge, wodurch der Arbeitsaufwand, der zum Abschließen des Schreibvorgangs erforderlich ist, vervielfacht wird.

Strategien zur Problembehebung

- Erstellen Sie bei Bedarf einen Index mit mehreren Schlüsseln und begrenzen Sie die Anzahl der Felder im Array.
- Beschränken Sie die Anzahl der Multi-Key-Indizes in der Sammlung
- Erwägen Sie, Ihr Datenmodell zu ändern

Tipps zur Leistungsverbesserung

Dieser Abschnitt enthält fünf Strategien zur Leistungsoptimierung für Amazon DocumentDB zur Verbesserung der Anwendungseffizienz und der Abfrageausführung.

1. Verwenden Sie $match als erste Phase in Aggregationspipelines

Platzieren Sie $match immer als erste Stufe für die Filterung in Ihrer Aggregationspipeline, um die Leistung zu maximieren. Amazon DocumentDB nutzt Indizes effektiv, wenn $match die Pipeline leitet, sodass die Datenbank Daten frühzeitig filtern und den Verarbeitungsaufwand reduzieren kann.

```
// Optimized approach
db.orders.aggregate([
  { $match: { status: "active", category: "electronics" } }, // Index utilization
  { $group: { _id: "$category", total: { $sum: "$price" } } },
  { $sort: { total: -1 } }
])
```

Auswirkung: Durch frühzeitiges Filtern wird die Anzahl der Dokumente reduziert, die in nachfolgenden Pipeline-Phasen verarbeitet werden, was zu einer schnelleren Abfrageausführung und einem geringeren Ressourcenverbrauch führt.

2. Verwenden Sie $project in der Aggregationspipeline, um die Größe der Pipeline-Daten zu minimieren

Führen Sie nur wichtige Felder durch die Phasen Ihrer Aggregationspipeline, um die Datengröße zu minimieren und die Leistung zu verbessern. Setzen Sie $project strategisch ein, um nur die Daten einzubeziehen, die Sie benötigen.

```
// Efficient pipeline design
db.orders.aggregate([
  { $match: { orderDate: { $gte: new Date("2024-01-01") } } },
  { $project: { customerId: 1, totalAmount: 1, status: 1 } }, // Only needed fields
  { $group: { _id: "$customerId", totalSpent: { $sum: "$totalAmount" } } }
])
```

Auswirkung: Kleinere Dokumente reduzieren den Speicherverbrauch und verbessern die Effizienz der Pipeline-Verarbeitung, was zu einer insgesamt verbesserten Abfrageleistung führt.

3. Aktivieren Sie die Dokumentenkomprimierung, um die Speicherkosten und Kosten zu I/O senken und die Abfrageleistung zu verbessern

Aktivieren Sie die Dokumentenkomprimierung anhand der Cluster-Parametergruppe, um Speicherkosten und Kosten zu I/O senken und die Abfrageleistung zu steigern. Amazon DocumentDB speichert komprimierte Dokumente sowohl auf der Festplatte als auch im RAM, wodurch der Speicherbedarf und die I/O Kosten reduziert werden.

Auswirkung:

- In den verfügbaren Speicher passen mehr Dokumente
- Schnellerer Datenzugriff mit weniger Festplattenlesevorgängen
- Niedrigere Speicherkosten und I/O Kosten und verbesserte Abfrageleistung

> **Note**
>
> Amazon DocumentDB aktiviert standardmäßig keine Komprimierung für Version 5.0. Sie können die Komprimierung auf Sammlungs- oder Clusterebene für 5.0-Cluster aktivieren. Verwenden Sie das Komprimierungsüberprüfungsprogramm von Amazon DocumentDB, um die Komprimierungsraten für Ihre Sammlungen zu analysieren.
> Für Amazon DocumentDB 8.0 ist die Komprimierung standardmäßig aktiviert.

4. Nutzen Sie Indizes für eine optimale Abfrageleistung

Stellen Sie sicher, dass Ihre Abfragen immer Indizes verwenden, um eine optimale Leistung zu erzielen. Amazon DocumentDB bietet mehrere Indextypen für unterschiedliche Anwendungsfälle.

Prinzipien der Indexierung:

- Jede Abfrage sollte einen geeigneten Index verwenden
- Amazon DocumentDB bietet mehrere Indextypen
- Zusammengesetzte Indizes bieten die größte Flexibilität, da sie verschiedene Abfrageformen mit einem einzigen Index unterstützen
- Entwerfen Sie Indizes so, dass sie Sortier- und Filtervorgänge gemeinsam unterstützen

Grundlegendes zu Indexpräfixen: Zusammengesetzte Indizes arbeiten mit Indexpräfixen — Amazon DocumentDB kann jede Teilmenge der Indexfelder verwenden, die von links nach rechts geschrieben werden. Der Index erstellt beispielsweise diese verwendbaren Präfixe: `{ category: 1, price: -1, inStock: 1 }`

- `{ category: 1 }`- unterstützt Abfragen, die nur nach Kategorien gefiltert werden
- `{ category: 1, price: -1 }`- unterstützt Abfragen, die nach Kategorie und Preis gefiltert sorting/filtering werden
- `{ category: 1, price: -1, inStock: 1 }`- unterstützt die vollständige zusammengesetzte Abfrage

Abfragen nach Preis, InStock oder InStock allein verwenden diesen Index nicht, da sie nicht mit dem ersten Feld (Kategorie) beginnen.

So identifizieren Sie Abfragen, die keine Indizes verwenden: Verwenden Sie die Methode explain (), um die Abfrageausführung zu analysieren und Abfragen zu identifizieren, die Sammelscans anstelle von Indizes durchführen.

Auswirkung: Abfragen ohne Indexauslastung führen zu Sammlungsscans, was zu einer erhöhten Speicher- und CPU-Auslastung der Instanz und zu einer erhöhten Abfragelatenz führt.

5. Optimieren Sie Datenmodelle auf der Grundlage von Abfragemustern

Passen Sie Ihr Datenmodell an die Art und Weise an, wie Ihre Anwendung Daten abfragt und aktualisiert. Datenmodellierung ist die Grundlage für leistungsstarke Amazon DocumentDB DocumentDB-Anwendungen.

Optimierungsstrategien:

Einbettung aus Gründen der Leistung

- Speichern Sie zusammengehörende Daten zusammen, wenn häufig auf sie zugegriffen wird, als Einheit
- Betten Sie Dokumente ein, die immer zusammen abgerufen werden
- Geeignet für persönliche Beziehungen

```
// Embedded approach for frequently accessed data
{
  _id: ObjectId("..."),
  customerName: "John Doe",
  address: {
    street: "123 Main St",
    city: "Seattle",
    zipCode: "98101"
  },
  recentOrders: [
    { orderId: "ORD001", amount: 99.99, date: "2024-01-15" }
  ]
}
```

Aus Gründen der Flexibilität Referenzierung

- Verwenden Sie Verweise für große oder selten abgerufene Daten
- Empfohlen für Eins-zu-Viele-Beziehungen mit großen Datensätzen
- Beugt aufgeblähten Dokumenten vor und verbessert die Aktualisierungsleistung

Strategie zur Aufteilung von Sammlungen

Wenn nur wenige Felder in großen Dokumenten häufig aktualisiert werden oder wenn große, selten aufgerufene Daten Dokumente überladen, sollten Sie eine Aufteilung von Sammlungen in Betracht ziehen:

- Bewahren Sie häufig aktualisierte Felder in einer separaten, kleineren Sammlung auf
- Speichern Sie statische Daten oder Daten, auf die selten zugegriffen wird, in einer anderen Sammlung
- Verknüpfen Sie sie bei Bedarf mit Referenzen

```
// Before: Large document with mixed access patterns
{
  _id: ObjectId("..."),
  productId: "PROD123",
  name: "Wireless Headphones",        // Frequently accessed
  price: 99.99,                       // Frequently accessed
  inventory: 45,                      // Updated frequently
  lastSold: "2024-01-15",           // Updated frequently
  detailedSpecs: { /* large object */ }, // Infrequently accessed
  manualPDF: "base64...",           // Large, rarely accessed
  reviewHistory: [/* large array */] // Infrequently accessed
}

// After: Split into collections based on access patterns
// products collection (frequently accessed data)
{
  _id: ObjectId("..."),
  productId: "PROD123",
  name: "Wireless Headphones",
  price: 99.99,
  inventory: 45,
  lastSold: "2024-01-15"
}

// product_details collection (infrequently accessed data)
{
  _id: ObjectId("..."),
  productId: "PROD123",           // Reference to products collection
  detailedSpecs: { /* large object */ },
  manualPDF: "base64...",
  reviewHistory: [/* large array */]
```

```
}
```

Leistungssteigerung: Kleinere Dokumente bedeuten schnellere Aktualisierungen, weniger Speicherverbrauch und verbesserte Cache-Effizienz.

Auswirkung: Ineffiziente Datenmodellierung führt zu suboptimalen Abfragen, erhöhten Dokumentengrößen und erhöhtem Speicherverbrauch, was zu einer verminderten Anwendungsleistung und höheren Betriebskosten führt.

Plananalyse abfragen

Die Analyse des Abfrageplans mithilfe von Explain Plan bietet wichtige Einblicke in die Abfrageleistung von Amazon DocumentDB. Der Abfrageplan mit executionStats zeigt wichtige Kennzahlen wie:

- Pro Phase zurückgegebene Dokumente (nReturned)
- Stage-specific Ausführungszeiten (AusführungTimeMillisEstimate)
- Dauer der Planerstellung (PlanungTimeMillis)

Durch die Untersuchung der Ergebnisse des Abfrageplans können Entwickler Ausführungsmuster analysieren, die Indexauslastung bewerten und potenzielle Optimierungsmöglichkeiten in allen Phasen der Abfragepipeline identifizieren.

Um den Abfrageplan zu analysieren, können Sie den Befehl explain () in den folgenden Formaten verwenden.

```
db.runCommand({explain: {query document}, verbosity: "executionStats"})
db.collection.find().explain("executionStats");
```

Im Folgenden finden Sie ein Beispiel für eine Operation:

```
db.collection.find({ companyname: { '$eq': 'ANYCOMPANY' }, isDeleted: { '$eq':
 false } }).sort({"createdAt":1}).limit(2).explain("executionStats");
```

Die Ausgabe dieser Operation sieht in etwa wie folgt aus:

```
{
  queryPlanner: {
    plannerVersion: 2,
    namespace: 'limit_test.test',
    winningPlan: {
      stage: 'LIMIT_SKIP',
      inputStage: {
        stage: 'SORT',
        sortPattern: { createdAt: 1 },
        inputStage: {
          stage: 'IXSCAN',
          indexName: 'companyname_1_createdAt_1_isDeleted_1',
          direction: 'forward',
          indexCond: {
            '$and': [
              { companyname: { '$eq': 'ANYCOMPANY' } },
              { isDeleted: { '$eq': false } }
            ]
          }
        }
      }
    }
  },
  indexFilterSet: false,
  indexFilterApplied: false,
  executionStats: {
    executionSuccess: true,
    executionTimeMillis: '4.186',
    planningTimeMillis: '3.909',
    executionStages: {
      stage: 'LIMIT_SKIP',
      nReturned: '2',
      executionTimeMillisEstimate: '0.199',
      inputStage: {
        stage: 'SORT',
        nReturned: '2',
        executionTimeMillisEstimate: '0.197',
        sortPattern: { createdAt: 1 },
        inputStage: {
          stage: 'IXSCAN',
          nReturned: '34',
```

```
          executionTimeMillisEstimate: '0.151',
          indexName: 'companyname_1_createdAt_1_isDeleted_1',
          direction: 'forward',
          indexCond: {
            '$and': [
              { companyname: { '$eq': 'ANYCOMPANY' } },
              { isDeleted: { '$eq': false } }
            ]
          }
        }
      }
    }
  },
  serverInfo: { host: 'demo-cluster', port: 27017, version: '5.0.0' },
  ok: 1,
  operationTime: Timestamp({ t: 1759915116, i: 1 })
}
```

Im Folgenden finden Sie eine detaillierte Analyse eines Amazon DocumentDB DocumentDB-Abfrageausführungsplans, in dem jede Komponente und ihre Leistungsmerkmale aufgeschlüsselt sind.

Allgemeiner Zeitplan

Die Ausführung TimeMillis stellt die Gesamtzeit dar, die für die Abfrage benötigt wurde, einschließlich der Planungszeit.

Planung TimeMillis stellt die gesamte Planungszeit dar, die für die Abfrage benötigt wurde.

Phasen der Ausführung

Es beschreibt Schritt für Schritt, wie Amazon DocumentDB eine Abfrage ausführt, und zeigt, wie Daten durch verschiedene Operationen fließen.

```
"executionStages": {
    "stage": "[STAGE_NAME]",
    "nReturned": "[NUMBER_OF_DOCS]",
    "executionTimeMillisEstimate": "[TIME]",
    "inputStage": {
        // Nested stages
    }
```

```
}
```

Allgemeine Phasen in einem Abfrageplan

Im Folgenden sind die häufigsten Ausführungsphasen in einem Abfrageplan aufgeführt. In jeder Phase werden Metriken für Ausführung TimeMillisEstimate (Ausführungszeit) und NReturned (Anzahl der Dokumente) zurückgegeben, mit deren Hilfe die Abfrageleistung in jeder Phase bewertet werden kann.

COLLSCAN (Sammlungsscan)

- Scannt die gesamte Sammlung Dokument für Dokument
- nReturned: Gibt alle passenden Dokumente in der Sammlung zurück
- Kann ein teurer Vorgang sein, wenn kein Index verfügbar ist

IXSCAN (Indexscan)

- Verwendet einen Index, um passende Dokumente zu finden
- Zurückgegeben: Nur übereinstimmende Dokumente, die auf dem Index basieren
- Effizienter Betrieb, gegenüber COLLSCAN bevorzugt

SORT

- Sortiert Dokumente anhand bestimmter Felder
- Wenn das Sortierattribut in der Abfrage nicht Teil eines Indexes ist, wird diese Phase explizit angezeigt
- Zurückgegeben: Dieselbe Zahl wie bei den Eingabedokumenten
- Memory-intensive für große Ergebnismengen. Fügen Sie zur Optimierung den Namen des Sortierfeldes einem Index hinzu

LIMIT_SKIP

- Steuert die Anzahl der zurückgegebenen und übersprungenen Dokumente
- Rückgegeben: Limitiert durch den LIMIT-Wert
- Wird für Paginierung oder LIMIT-Operationen verwendet

SUBSCAN

- Führt verschachtelte Abfrageoperationen aus
- Zurückgegeben: Variiert je nach den Ergebnissen der Unterabfrage
- Wird in komplexen Abfragen mit mehreren Phasen verwendet

Phasen mit hoher Ausführungsrate TimeMillisEstimate eignen sich gut für eine Optimierung.

> **Note**
>
> Der ExecutionStats-Parameter unterstützt derzeit keine Aktualisierungs- und Löschbefehle.

Grundlegendes zu DocsExamined in Explain Plan Execution Stats

Wenn Sie eine Abfrage mit ausführen`explain("executionStats")`, bietet Amazon DocumentDB Prüfungsmetriken, anhand derer Sie nachvollziehen können, wie viele Dokumente gescannt wurden, um die Abfrageergebnisse zu erhalten. Diese Metriken sind nützlich, um ineffiziente Abfragepläne zu identifizieren und die Leistung zu optimieren.

> **Note**
>
> Nur in Amazon DocumentDB 5.0.1+ und 8.0.0+ verfügbar.

Felder

Feld	Description	Level
insgesamt DocsExamined	Gesamtzahl der Dokumente, die in allen Ausführungsphasen geprüft wurden.	Top-level execution Stats
Dokumente geprüft	Anzahl der Dokumente, die in einer bestimmten Ausführungsphase geprüft wurden.	Stage-level

Das `docsExamined` Feld erscheint in den folgenden Phasen:

Stage	Description
COLLSCAN	Sammlung scannen. Alle Dokumente in der Sammlung werden geprüft.
ISCAN	Indexscan. Es werden nur Dokumente untersucht, die der Indexbedi ngung entsprechen.
FETCH	Wenn der Optimierer Dokumente in einer separaten Phase von IXSCAN abruft, meldet die FETCH-Phase den Wert DocsExamined. In Indexscan plänen melden die untergeordneten IXSCAN-Stufen nicht den Wert docsExamined. In $lookup-Plänen können sowohl die FETCH-Phase als auch ihre untergeordneten Stufen den Wert DocsExamined melden.

> **Note**
>
> `docsExamined`wird in den Stufen IXONLYSCAN nicht angezeigt, da die Abfrage vollständig aus dem Index beantwortet wird, ohne dass auf Dokumente zugegriffen wird.

In der Ausgabe von Explain Plan executionStats Dokumente anzeigen

Die folgenden Beispiele verwenden eine Sammlung mit 500.000 Dokumenten und veranschaulichen, wie sich die `docsExamined` Änderungen in den verschiedenen Phasen der Abfrageausführung ändern.

Erstellen Sie Beispieldaten:

```
for (let i = 0; i < 500000; i++) {
    db.coll.insertOne({ number: i, arr: [i, [i+1]], value: "test", bool: i % 2 === 
 0 });
}
```

Sammlungsscan (COLLSCAN)

Ohne Index führt Amazon DocumentDB einen vollständigen Sammlungsscan durch.

```
db.coll.find({ "number": { "$lt": 500 } }).explain("executionStats").executionStats
```

Ausgabe:

```
{
    "executionSuccess" : true,
    "nReturned" : "500",
    "executionTimeMillis" : "282.055",
    "planningTimeMillis" : "0.085",
    "totalDocsExamined" : "500000",
    "executionStages" : {
        "stage" : "COLLSCAN",
        "nReturned" : "500",
        "executionTimeMillisEstimate" : "281.915",
        "docsExamined" : "500000",
        "filter" : {
            "number" : {
                "$lt" : 500
            }
        }
    }
}
```

Alle 500.000 Dokumente wurden untersucht und lieferten 500 Ergebnisse. Das Erstellen eines Indexes für das Zahlenfeld verbessert dies.

Indexscan (IXSCAN)

```
db.coll.createIndex({ number: 1 });
db.coll.find({ "number": { "$lt": 5000 } }).explain("executionStats").executionStats
```

Ausgabe:

```
{
    "executionSuccess" : true,
    "nReturned" : "5000",
```

```
    "executionTimeMillis" : "3.047",
    "planningTimeMillis" : "0.296",
    "totalDocsExamined" : "5000",
    "executionStages" : {
        "stage" : "IXSCAN",
        "nReturned" : "5000",
        "executionTimeMillisEstimate" : "2.576",
        "indexName" : "number_1",
        "direction" : "forward",
        "docsExamined" : "5000",
        "indexCond" : {
            "$and" : [
                {
                    "number" : {
                        "$lt" : 5000
                    }
                }
            ]
        }
    }
}
```

Mit dem Index wurden nur 5.000 von 500.000 Dokumenten geprüft und abgerufen, was der zurückgegebenen Zahl entsprach. Durch die Indexbedingung wurden 495.000 Dokumente herausgefiltert, die der Abfrage nicht entsprachen.

Indexscan mit Restfilter

```
db.coll.find({ "number": { "$lt": 5000 }, "arr": { "$gt":
 4000 } }).explain("executionStats").executionStats
```

Ausgabe:

```
{
    "executionSuccess" : true,
    "nReturned" : "999",
    "executionTimeMillis" : "15.367",
    "planningTimeMillis" : "0.115",
    "totalDocsExamined" : "5000",
```

```
    "executionStages" : {
        "stage" : "IXSCAN",
        "nReturned" : "999",
        "executionTimeMillisEstimate" : "15.170",
        "indexName" : "number_1",
        "direction" : "forward",
        "docsExamined" : "5000",
        "indexCond" : {
            "$and" : [
                {
                    "number" : {
                        "$lt" : 5000
                    }
                }
            ]
        },
        "filter" : {
            "arr" : {
                "$gt" : 4000
            }
        }
    }
}
```

Die Indexbedingung entsprach 5.000 von 500.000 Dokumenten, danach `arr` reduzierte der Restfilter das Ergebnis auf 999. Der `docsExamined` Wert 5.000 spiegelt alle Dokumente wider, die nach der Indexbedingung, aber vor der Anwendung des Restfilters untersucht wurden.

Mit IXSCAN abrufen

```
db.coll.find({ "$or": [{ "number": { "$lt": 100000 } }, { "number": { "$gt":
 400000 } }] }).explain("executionStats").executionStats
```

Ausgabe:

```
{
    "executionSuccess" : true,
    "nReturned" : "199999",
    "executionTimeMillis" : "899.801",
```

```
    "planningTimeMillis" : "0.183",
    "totalDocsExamined" : "199999",
    "executionStages" : {
        "stage" : "FETCH",
        "nReturned" : "199999",
        "executionTimeMillisEstimate" : "894.141",
        "docsExamined" : "199999",
        "inputStage" : {
            "stage" : "IXOR",
            "nReturned" : "0",
            "executionTimeMillisEstimate" : "874.897",
            "inputStages" : [
                {
                    "stage" : "IXSCAN",
                    "nReturned" : "100000",
                    "executionTimeMillisEstimate" : "462.208",
                    "indexName" : "number_1",
                    "indexCond" : {
                        "$and" : [
                            {
                                "number" : {
                                    "$lt" : 100000
                                }
                            }
                        ]
                    }
                },
                {
                    "stage" : "IXSCAN",
                    "nReturned" : "99999",
                    "executionTimeMillisEstimate" : "412.684",
                    "indexName" : "number_1",
                    "indexCond" : {
                        "$and" : [
                            {
                                "number" : {
                                    "$gt" : 400000
                                }
                            }
                        ]
                    }
                }
            ]
        }
```

```
    }
}
```

Wenn der Amazon DocumentDB DocumentDB-Optimierer eine Abrufphase zum Abrufen von Dokumenten verwendet, werden in der FETCH-Phase Berichte erstellt. `docsExamined` Die untergeordneten IXSCAN-Stufen erstellen keine Berichte, `docsExamined` da sie nur Indexschlüssel scannen, ohne direkt auf Dokumente zuzugreifen.

FETCH mit aggregierter Suche

```
db.coll.explain("executionStats").aggregate([
    { $match: { "number": { "$lt": 5 } } },
    { $lookup: { from: "coll", pipeline: [{ $match: { "number": { "$lt": 3 } } }], as:
 "sub" } }
]).executionStats
```

Ausgabe:

```
{
    "executionSuccess" : true,
    "nReturned" : "5",
    "executionTimeMillis" : "0.525",
    "planningTimeMillis" : "0.327",
    "totalDocsExamined" : "9",
    "executionStages" : {
        "stage" : "NESTED_LOOP_LOOKUP",
        "nReturned" : "5",
        "executionTimeMillisEstimate" : "0.163",
        "inputStages" : [
            {
                "stage" : "IXSCAN",
                "nReturned" : "5",
                "executionTimeMillisEstimate" : "0.039",
                "indexName" : "number_1",
                "direction" : "forward",
                "docsExamined" : "5",
                "indexCond" : {
                    "$and" : [
```

```
                        {
                            "number" : {
                                "$lt" : 5
                            }
                        }
                    ]
                }
            },
            {
                "stage" : "FETCH",
                "nReturned" : "1",
                "executionTimeMillisEstimate" : "0.009",
                "docsExamined" : "1",
                "inputStage" : {
                    "stage" : "AGGREGATE",
                    "nReturned" : "1",
                    "executionTimeMillisEstimate" : "0.044",
                    "inputStage" : {
                        "stage" : "IXSCAN",
                        "nReturned" : "3",
                        "executionTimeMillisEstimate" : "0.013",
                        "indexName" : "number_1",
                        "direction" : "forward",
                        "docsExamined" : "3",
                        "indexCond" : {
                            "$and" : [
                                {
                                    "number" : {
                                        "$lt" : 3
                                    }
                                }
                            ]
                        }
                    }
                }
            }
        ]
    }
}
```

Der äußere IXSCAN untersuchte 5 Dokumente, die der Zahl < 5 entsprachen. Der innere IXSCAN untersuchte 3 Dokumente, die der Zahl < 3 entsprachen, und in der FETCH-Phase wurde ein

aggregiertes Ergebnis untersucht. Die `totalDocsExamined` Zahl 9 ist die Summe aller Stufen (5 + 3 + 1).

Index-only Scans (abgedeckte Abfragen)

Ein reiner Index-Scan (auch als abgedeckte Abfrage bezeichnet) ist eine Optimierung, bei der Amazon DocumentDB eine Abfrage ausschließlich anhand der in einem Index gespeicherten Felder beantwortet, ohne die zugrunde liegenden Dokumente zu lesen. Da die Werte, die eine Abfrage benötigt, bereits im Index vorhanden sind, kann Amazon DocumentDB das Abrufen jedes passenden Dokuments überspringen, wodurch die zufälligen I/O und dekodierenden Kosten für das Lesen vollständiger Dokumente vermieden werden. Im Abfrageplan wird ein reiner Index-Scan als `IXONLYSCAN` Phase anstelle eines `IXSCAN` (Index-Scans) oder `COLLSCAN` (Sammlungsscans) angezeigt. Dadurch wird die Abfragelatenz für unterstützte Abfragen reduziert I/O und kann diese erheblich verringern.

Eine Abfrage eignet sich gut für einen reinen Index-Scan, wenn sie nur eine kleine Anzahl von Feldern betrifft, die alle in einem einzigen Index vorhanden sind, insbesondere wenn die Dokumente selbst groß sind. Gängige Beispiele sind Dashboards, Analytics-Rollups und Lookups, bei denen einige Felder aus einem größeren zusammengesetzten Index gelesen werden.

Die Ergebnisse einer Abfrage sind dieselben, unabhängig davon, ob ein reiner Index-Scan verwendet wird oder nicht. Nur der Abfrageplan und der Umfang, in dem I/O er ausgeführt wird, ändern sich.

> **Note**
>
> Index-only Die Scanunterstützung hängt von Ihrer Amazon DocumentDB-Engine-Version ab:
>
> - Für abgedeckte `find` Prognosen, die mehr als ein Feld enthalten, das kein `_id` Feld ist, ist die Engine-Version 8.0.2 oder höher erforderlich. In früheren Versionen kann eine abgedeckte `find` Projektion höchstens ein Feld enthalten, das kein `_id` Feld ist. Eine Projektion mit zwei oder mehr Feldern fällt auf einen normalen Index- oder Sammelscan zurück.
> - Index-only Für Scans nach Aggregations-Pipelines, z. B. für Covered`$group`, ist die Engine-Version 8.0.2 oder höher erforderlich.

Wenn Amazon DocumentDB einen reinen Index-Scan verwendet

Amazon DocumentDB erwägt für die folgenden Operationen einen reinen Index-Scan, wenn ein einziger Index jedes Feld enthält, das der Vorgang liest.

Abgedeckt, Prognosen finden

Eine `find` Abfrage ist qualifiziert, wenn ihre Projektion die einfache Einbeziehung eines oder mehrerer Felder der obersten Ebene ist und jedes projizierte Feld in einem einzigen Index enthalten ist. Wenn beispielsweise ein zusammengesetzter Index aktiviert ist`{ customerId: 1, status: 1, total: 1 }`, wird die folgende Abfrage behandelt. Alle drei projizierten Felder befinden sich im Index und `_id` sind ausdrücklich ausgeschlossen. (Um eine Projektion mit mehr als einem `_id` Nicht-Feld abzudecken, ist Engine-Version 8.0.2 oder höher erforderlich.)

```
db.orders.createIndex({ customerId: 1, status: 1, total: 1 })

db.orders.find(
  { customerId: 42 },
  { _id: 0, customerId: 1, status: 1, total: 1 }
)
```

Der erfolgreiche Plan verwendet eine `IXONLYSCAN` Stufe und holt niemals die passenden Dokumente aus dem Speicher ab:

```
{
  "stage": "IXONLYSCAN",
  "indexName": "customerId_1_status_1_total_1",
  "direction": "forward",
  "indexCond": { "$and": [ { "customerId": { "$eq": 42 } } ] }
}
```

> **Note**
>
> In einer abgedeckten `find` Abfrage werden die Felder im Ergebnis in der Indexreihenfolge zurückgegeben (in der Reihenfolge, in der die Felder in der Indexdefinition erscheinen), nicht

in der Reihenfolge, in der sie in der Projektion aufgeführt wurden, oder in der Reihenfolge, in der sie im Dokument gespeichert sind. Dies entspricht dem Verhalten von MongoDB für abgedeckte Abfragen. Eine Abfrage, die durch einen Sammlungsscan beantwortet wird, gibt stattdessen Felder in der Reihenfolge der gespeicherten Dokumente zurück.

Es wurden $group-Aggregationen abgedeckt

Eine `$group` Phase liest normalerweise nur wenige Felder (den Gruppenschlüssel und die Eingaben für ihre Akkumulatoren), wird aber oft ohne vorangestelltes Zeichen ausgeführt. `$match` Ohne einen übergeordneten Index liest und dekodiert Amazon DocumentDB jedes vollständige Dokument in der Sammlung, nur um diese wenigen Felder in die Gruppierung einzufügen.

Index-only Scans für Aggregationspipelines, einschließlich der abgedeckten `$group` Stufen, sind in der Engine-Version 8.0.2 und höher verfügbar.

Wenn ein einzelner Index jedes Feld enthält, das das Pipeline-Präfix liest, kann Amazon DocumentDB einen reinen Index-Scan unter der Gruppe platzieren und diese Felder direkt aus dem Index lesen. Stellen Sie sich eine Pipeline vor, die Bestellungen nach Status gruppiert und ihre Gesamtsummen summiert, mit einem zusammengesetzten Index für: `{ status: 1, total: 1 }`

```
db.orders.createIndex({ status: 1, total: 1 })

db.orders.aggregate([
  { $group: { _id: "$status", revenue: { $sum: "$total" } } }
])
```

Die Pipeline liest nur `status` und `total` beide sind im Index enthalten, sodass die Gruppe nicht aus einem Sammelscan, `IXONLYSCAN` sondern aus einem Sammelscan gescannt wird:

```
// Without a covering index
$group  <-  COLLSCAN     (reads and decodes every full document)

// With a covering index on { status, total }
$group  <-  IXONLYSCAN over { status, total }   (reads only the referenced fields)
```

Ein führendes `$match``$sort`,`$skip`, oder `$limit` vor dem `$group` wird unterstützt. Jedes Feld, auf das diese Stufen verweisen (z. B. ein `$match` Filterfeld oder ein `$sort` Schlüssel), muss jedoch auch im gleichen Index vorhanden sein, damit die Pipeline abgedeckt wird. Wenn ein Feld, auf das verwiesen wird, im Index fehlt, greift Amazon DocumentDB auf einen Sammlungs- oder Index-Scan zurück.

> **Note**
>
> Im Gegensatz zu einer `find` Projektion `$group` schließt a das Dokument nicht implizit ein. `_id` `_id`In a `$group` ist der Gruppenschlüssel, nicht die Dokument-ID, sodass der Index das Dokument `_id` nicht enthalten muss, um die Pipeline abzudecken.

Anforderungen für eine abgedeckte Abfrage

Damit Amazon DocumentDB einen reinen Index-Scan verwenden kann, müssen alle folgenden Bedingungen erfüllt sein:

- Ein einziger Index enthält jedes Feld, das die Abfrage liest. Dazu gehören projizierte Felder für `find` und den Gruppenschlüssel, Akkumulatoreingaben und alle Filter- oder Sortierfelder für eine abgedeckte `$group` Pipeline. Wenn auch nur ein Feld im Index fehlt, kann die Abfrage nicht abgedeckt werden.
- Die Projektion beinhaltet Felder der obersten Ebene (für`find`). Ausschlussprognosen, berechnete Felder und Projektionsoperatoren wie `$slice` und `$elemMatch` sind nicht zulässig.
- Das **`_id`** Feld wird berücksichtigt (für`find`). Weil standardmäßig enthalten `_id` ist, ist ein allgemeiner zusammengesetzter Index wie „Nicht *`{ a: 1, b: 1 }`* deckt“`find({}, { a: 1, b: 1 })`: Das Implizite `_id` ist nicht im Index enthalten. Entweder `_id` aus der Projektion ausschließen (`{ _id: 0, a: 1, b: 1 }`) oder `_id` zum Index hinzufügen.
- Der Filter entspricht voll und ganz dem Index. Wenn Amazon DocumentDB einen Teil des Abfrageprädikats nicht in eine Indexbindung umwandeln kann, behält es diesen Teil als Restfilter bei, liest das Dokument, um es auszuwerten, und konvertiert den Scan nicht in einen reinen Index-Scan. Operatoren wie`$mod`, `$ne``$nin`, und `$exists` hinterlassen einen Restfilter.
- Kein projiziertes Feld ist eine Mehrschlüsselspalte (Array). Siehe Einschränkungen.

Einschränkungen

In den folgenden Fällen kann ein reiner Index-Scan nicht verwendet werden und es wird auf einen normalen Index- oder Sammlungsscan zurückgegriffen. Dies ist ein erwartetes Verhalten und hat keinen Einfluss auf die Richtigkeit der Ergebnisse.

- Felder mit mehreren Schlüsseln (Array). Wenn ein Feld in einem Dokument jemals einen Array-Wert enthielt, ist seine Indexspalte mit mehreren Schlüsseln versehen. Ein Mehrschlüsselindex speichert einen Eintrag pro Array-Element, sodass das ursprüngliche Array nicht aus dem Index rekonstruiert werden kann. Jede Abfrage, die ein solches Feld liest, wird nicht behandelt.
- Gepunktete oder verschachtelte Pfade. Eine Projektion oder ein Feldreferenz auf einem verschachtelten Pfad wie `"address.city"` ist nicht zulässig, auch wenn der Index diesen Pfad enthält.
- Ausschluss-, berechnete oder Operatorprojektionen. Prognosen wie`{ b: 0 }`,`{ total: { $add: ["$price", "$tax"] } }`, oder `{ a: { $slice: 5 } }` erfordern das vollständige Dokument und werden nicht behandelt.
- Whole-document Referenzen in Aggregation. Ein Dokument`$group`, das auf das gesamte Dokument verweist (z. B.`{ $push: "$$ROOT" }`), kann nicht abgedeckt werden, da kein Index das gesamte Dokument rekonstruieren kann.
- Vorangegangene Etappen umgestalten. **`$group`** Eine Phase, in der Dokumente (wie`$project`,, oder`$lookup`)`$addFields`, die vor dem erscheinen`$unwind`, neu gestaltet werden, verhindert`$group`, dass die Pipeline abgedeckt wird.
- Die Sortierung stimmt bei Zeichenkettenprädikaten nicht überein. Wenn sich die Sortierung der Abfrage von der Sortierung des Indexes unterscheidet, können Zeichenkettenprädikate nicht vom Index bereitgestellt werden und hinterlassen einen Restfilter, der einen Dokumentabruf erzwingt. Numerische und typbasierte Prädikate sind davon nicht betroffen.

Bestätigung eines reinen Index-Scans

Bestätigen Sie anhand der `explain("executionStats")` Ausgabe, ob eine Abfrage nur durch einen Indexscan bearbeitet wird und dass Dokumente nicht betroffen sind. Achten Sie auf zwei Dinge:

- Der Gewinnplan nutzt die `IXONLYSCAN` Bühne.

- Der `docsExamined` Wert in dieser Phase ist `0` (und `totalDocsExamined` ist`0`), was bedeutet, dass die Abfrage vollständig aus dem Index heraus beantwortet wurde, ohne dass Dokumente gelesen wurden.

```
db.orders.find(
  { customerId: 42 },
  { _id: 0, customerId: 1, status: 1, total: 1 }
).explain("executionStats").executionStats
```

```
{
    "executionSuccess": true,
    "nReturned": "2",
    "executionTimeMillis": "0.714",
    "totalDocsExamined": "0",
    "executionStages": {
        "stage": "IXONLYSCAN",
        "nReturned": "2",
        "indexName": "customerId_1_status_1_total_1",
        "direction": "forward",
        "docsExamined": "0"
    }
}
```

> **Note**
>
> Ein `docsExamined` Wert ungleich Null auf einer `IXONLYSCAN` Stufe bedeutet, dass der Index die Daten enthielt, aber Amazon DocumentDB überprüfte trotzdem den Speicher, um zu bestätigen, dass jede Dokumentversion sichtbar war. Dies kann passieren, wenn die Sammlung kürzlich Schreibvorgänge enthält, deren veraltete Dokumentversionen die Garbage Collection noch nicht abgerufen hat. Sobald die Garbage Collection diese Dokumente verarbeitet hat, `docsExamined` fällt sie auf. `0` An diesem Punkt realisiert die Abfrage den vollen Nutzen des abgedeckten Scans. Weitere Informationen zur Speicherbereinigung finden Sie unterMüllabfuhr in Amazon DocumentDB. Für weitere Informationen über die `docsExamined`-Metrik siehe Plananalyse abfragen.

Bewährte Methoden

- Entwerfen Sie zusammengesetzte Indizes, die die Felder enthalten, nach denen Ihre häufigsten Abfragen projiziert oder gruppiert werden, sodass diese Abfragen abgedeckt werden können.
- `_id`Aus `find` Projektionen ausschließen (`{ _id: 0, ... }`), sofern es nicht Teil des abdeckenden Indexes `_id` ist.
- Vermeiden Sie es, Array-Felder (Mehrschlüssel) zu indizieren, wenn Sie beabsichtigen, eine Abfrage abzudecken, da Mehrschlüsselspalten den reinen Indexpfad disqualifizieren.
- Sorgen Sie dafür, dass die Garbage-Collection mit Ihrem Schreibaufwand Schritt hält, sodass abgedeckte Scans den Zugriff auf Dokumente überspringen und konstant bleiben. `docsExamined 0` Weitere Informationen finden Sie unter Müllabfuhr in Amazon DocumentDB.

Cache-Filter planen

Ein Plan-Cache-Filter, auch Indexfilter genannt, schränkt die Menge der Indizes ein, die der Abfrageplaner für eine bestimmte Abfrageform berücksichtigen darf. Wenn eine Abfrage mit einem Shape übereinstimmt, das über einen Filter verfügt, wählt der Planer einen Plan nur aus den in diesem Filter genannten Indizes aus, anstatt jeden Index in der Sammlung auszuwerten.

Ein Abfrage-Shape ist die Kombination aus dem Abfrageprädikat, der Sortierspezifikation und dem Sammlungsnamespace, wobei die Prädikatwerte normalisiert sind. Zwei Abfragen, die sich nur in ihren Werten unterscheiden, haben eine gemeinsame Form, sodass ein einziger Filter sie alle abdeckt. In der Ausgabe von `planCacheListFilters` werden normalisierte Werte als `"@"` angezeigt.

Mit einem Plan-Cache-Filter können Sie den Plan fixieren, sodass Sie vermeiden müssen, aufgrund der Datenverteilung oder Indexänderungen einen langsameren Index auszuwählen. Filter haben die folgenden Eigenschaften:

- Es ist keine Änderung der Anwendung erforderlich. Filter werden mit einem Datenbankbefehl festgelegt und auf dem Server angewendet, sodass Sie eine Regression ohne Codebereitstellung abschwächen können. Dies ist der Hauptunterschied zu einem`hint`, das an jeder Anrufstelle hinzugefügt werden muss.
- Der Gültigkeitsbereich ist ein Abfrage-Shape. Eine Form wird durch die Sammlung, das Prädikat und die Sortierung definiert, nicht durch die vollständige Abfrage, sodass ein Filter nur die Abfragen einschränkt, die dieser Form entsprechen. Andere Abfragen in derselben Sammlung werden weiterhin normal geplant.

- Die Änderung ist dauerhaft. Filter bleiben auch bei Instanzneustarts und Engine-Patches bestehen. Wenn Sie die Sammlung löschen, werden ihre Filter entfernt.
- Die Änderung ist reversibel. Wenn Sie den Filter löschen, kehrt die Form zur normalen kostenbasierten Planung zurück. Ein Filter ist also eine risikoarme Methode, um eine Arbeitslast zu stabilisieren, während Sie die zugrunde liegende Ursache untersuchen.

Themen

- Unterstützte Befehle
- Unterstützte Abfrage-Shapes
- Filter setzen, auflisten und löschen
- Stellen Sie sicher, dass ein Filter angewendet wird
- Beispiele
- Verhalten bei Hinweisen und Indexänderungen

Unterstützte Befehle

Für Plan-Cache-Filter ist Planner-Version 2.0 oder höher erforderlich. Der Befehl, auf den ein Filter angewendet werden kann, hängt von der Planner-Version und der Amazon DocumentDB-Nebenversion ab, wie in der folgenden Tabelle dargestellt.

Befehlsunterstützung für Plan-Cache-Filter

Befehl	Minimale Planerversion	Minimale Nebenversion
`find`	2.0	5.0.0, 8.0.0
`count`	2.0	5.0.0, 8.0.0
`update`	2.0	5.0.2, 8.0.2
`delete`	2.0	5.0.2, 8.0.2
`findAndModify`	2.0	5.0.2, 8.0.2
`distinct`	3.0	8.0.0
`aggregate`	3.0	8.0.2

> **Note**
>
> Planner Version 3.0 ist nur in Amazon DocumentDB 8.0 verfügbar. Da für die `aggregate` Befehle `distinct` und die Planner-Version 3.0 erforderlich ist, sind Plan-Cache-Filter für diese beiden Befehle nur in Amazon DocumentDB 8.0 verfügbar. In Amazon DocumentDB 5.0, das Planner-Version 2.0 unterstützt, gelten Filter für die `count` Befehle`find`, `updatedelete`, und`findAndModify`.

Für den `aggregate` Befehl wird die Form von der Vorderseite der Pipeline übernommen: Die erste Seite `$match` versorgt den Filter und eine `$sort` unmittelbar danach liegende Leitung versorgt die Sortierung. `$skip`und wirken sich `$limit` nicht auf die Form aus, ebenso wenig wie die Stufen hinter diesem Punkt.

Amazon DocumentDB optimiert die Pipeline vor der Planung, und ein Plan-Cache-Filter wird mit der optimierten Pipeline abgeglichen. Der Filter und die Sortierung, die der Planer sieht, können sich daher von den Phasen unterscheiden, die Sie geschrieben haben. Beispielsweise werden zwei benachbarte `$match` Stufen an der Vorderseite einer Pipeline zu einem einzigen `$and` Prädikat zusammengefasst:

```
db.orders.aggregate([
  { $match: { status: "SHIPPED" } },
  { $match: { region: "us-east-1" } },
  { $group: { _id: "$customerId", total: { $sum: "$amount" } } }
])
```

Der Planer sieht ein `$match {$and: [{status: ...}, {region: ...}]}` Prädikat aktiviert, daher muss der Filter auf dieses kombinierte Prädikat gesetzt werden.

```
db.runCommand({planCacheSetFilter: "orders",
query: { $and: [ { status: "SHIPPED" }, { region: "us-east-1" } ] },
indexes: [ "status_1_region_1" ]})
```

Ein Filter, der für die ausländische Sammlung einer `$lookup` `$graphLookup` OR-Stufe gesetzt wird, steuert den Index, der zum Scannen dieser fremden Sammlung verwendet wird.

Unterstützte Abfrage-Shapes

Ab Amazon DocumentDB 5.0.2 und 8.0.2 können die folgenden Operatoren in einem Abfrage-Shape erscheinen, das über einen Plan-Cache-Filter verfügt:

- `$text`. Die Suchzeichenfolge ist normalisiert, sodass Abfragen, die sich nur in den Begriffen unterscheiden, nach denen sie suchen, eine gemeinsame Form haben.
- `$near` und `$nearSphere`. Beide Operatoren normalisieren sich auf dieselbe Form. Die Koordinaten`$minDistance`, und `$maxDistance` sind nicht Teil der Form.
- `$geoWithin` und `$geoIntersects`. Der GeoJSON-Geometrietyp ist Teil der Form, sodass eine `Polygon` Abfrage und eine `MultiPolygon` Abfrage unterschiedliche Formen haben. Die Koordinaten sind nicht Teil der Form.
- `$regex`, wenn es direkt auf einem Feld verwendet wird. Das Muster und die Optionen sind normalisiert, sodass alle regulären Ausdrücke im gleichen Feld dieselbe Form haben.

Das Setzen eines Filters schlägt mit dem Fehlercode 303 für ein Abfrage-Shape fehl, das `$jsonSchema``$expr`, oder `$sampleRate` verwendet. Es schlägt auch bei den folgenden Formen fehl. Eine Abfrage, die einen von ihnen verwendet, ist ohne Filter geplant:

- Ein darin `$regex` verschachteltes `$in``$nin`, oder`$all`. Ein Array, das reguläre Ausdrücke mit Literalwerten mischt, kann nicht anhand eines Indexes ausgewertet werden. Daher könnte niemals ein Filter auf diese Form angewendet werden.
- Eine Art Spezifikation von. `{$natural: 1}` Eine `$natural` Sortierung fordert einen Sammlungsscan an, was mit der Beschränkung des Planers auf einen Index nicht vereinbar ist.

Filter setzen, auflisten und löschen

Verwenden Sie den `planCacheSetFilter` Befehl, um einen Plan-Cache-Filter einzurichten. Die `sort` Felder `query` und geben zusammen das Abfrage-Shape an und `indexes` listen die Indexnamen auf, die der Planer für dieses Shape berücksichtigen darf.

```
db.runCommand({
planCacheSetFilter: <collection>,
query: <query>,
sort: <sort>, // optional
indexes: [ <index1>, <index2>, ...],
comment: <any> // optional
```

```
})
```

Beispielsweise beschränkt der folgende Befehl den Planer auf den a_1 Index für die Form `{a: {$eq: "@"}, b: {$eq: "@"}}` ohne Sortierung:

```
db.runCommand({planCacheSetFilter: "foo",
query: { a: 1, b: 1 },
sort: {},
indexes: [ "a_1" ]})
```

Wenn Sie einen Filter für eine Form festlegen, für die bereits ein Filter vorhanden ist, wird der vorhandene Filter ersetzt. Nachdem Sie die folgenden beiden Befehle ausgeführt haben, `{a: {$eq: "@"}, b: {$eq: "@"}}` lautet der Filter für die Form`["b_1"]`:

```
db.runCommand({planCacheSetFilter: "foo",
query: { a: 1, b: 1 },
sort: {},
indexes: [ "a_1" ]})

db.runCommand({planCacheSetFilter: "foo",
query: { a: 6, b: 10 },
sort: {},
indexes: [ "b_1" ]})
```

Verwenden Sie den `planCacheListFilters` folgenden Befehl, um alle Filter einer Sammlung aufzulisten:

```
db.runCommand({planCacheListFilters: <collection>})
```

Die Ausgabe zeigt jede gespeicherte Form mit ihren normalisierten Werten und der Liste der zulässigen Indizes:

```
{
"filters" : [
{
"query" : { "a" : { "$eq" : "@" } },
"sort" : { },
"indexes" : [ "a_1" ]
},
{
```

```
"query" : { "a" : { "$gt" : "@" } },
"sort" : { "a" : 1 },
"indexes" : [ "a_1_b_1" ]
}
],
"ok" : 1
}
```

Der Operator wird beibehalten und nur der Wert wird durch ersetzt`"@"`. Eine implizite Gleichheit wie `{a: 1}` wird in ihrer expliziten Form aufgeführt`{"a": {"$eq": "@"}}`, und die Elemente eines `$in` Arrays werden als Ganzes ersetzt, als`{"a": {"$in": "@"}}`. Ein Filtersatz ohne Sortierung wird mit einem leeren `sort` Dokument aufgeführt.

Verwenden Sie den `planCacheClearFilters` Befehl, um Filter zu entfernen.

```
db.runCommand({
planCacheClearFilters: <collection>,
query: <query pattern>, // optional
sort: <sort specification>, // optional
comment: <any> // optional
})
```

Um alle Filter in der Sammlung zu löschen, lassen Sie beide weg `query` und`sort`:

```
db.runCommand({planCacheClearFilters: "foo"})
```

Wenn Sie beide `query` angeben, `sort` wird nur der Filter gelöscht, der genau diese Sortierung hat. Die Werte in `query` sind normalisiert, sodass jeder repräsentative Wert funktioniert. Um nur den Filter zu löschen, der ohne Sortierung gesetzt wurde, übergeben Sie ein leeres `sort` Dokument:

```
db.runCommand({planCacheClearFilters: "foo", query: {a: 1}, sort: {a: 1}})

db.runCommand({planCacheClearFilters: "foo", query: {a: 1}, sort: {}})
```

Stellen Sie sicher, dass ein Filter angewendet wird

Die `explain` Ausgabe enthält zwei Felder, die den Status des Plan-Cache-Filters melden:

- `indexFilterSet`liegt vor`true`, wenn in der Sammlung ein Filter für die Form der Abfrage vorhanden ist.

- `indexFilterAppliedist` `true` nur dann der Fall, wenn die Abfrage anhand eines Indexes aus der Liste dieses Filters geplant wurde.

Ab Amazon DocumentDB 5.0.2 und 8.0.2 werden diese Felder für jeden Befehl gemeldet, der Plan-Cache-Filter unterstützt, und in der Profiler-Ausgabe für diese Befehle. In früheren Nebenversionen wurden sie nur für den Befehl gemeldet. `find` Weitere Hinweise zum Lesen der Explain-Ausgabe finden Sie unterPlananalyse abfragen.

Ein Filter kann einer Abfrageform entsprechen, ohne angewendet zu werden. Wenn keiner der Indizes in der Liste des Filters die Abfrage verarbeiten kann, plant der Planer ohne den Filter neu und wählt einen Kostenindex aus. In diesem Fall `indexFilterSet` ist `true` und `indexFilterApplied` ist. `false` Dies passiert, wenn die benannten Indizes die Felder der Abfrage nicht abdecken, wenn die benannten Indizes nicht existieren oder wenn eine `$regex` Form keine Indexgrenzen erzeugen kann, z. B. ein Muster ohne Verankerung oder ohne Berücksichtigung der Groß- und Kleinschreibung.

Beispiele

In den folgenden Beispielen wird eine `orders` Sammlung und für das Beispiel eine `customers` Sammlung mit diesen Indizes verwendet: `$lookup`

```
db.orders.createIndex({ status: 1 })                          // status_1
db.orders.createIndex({ status: 1, orderDate: 1 })            // status_1_orderDate_1
db.orders.createIndex({ customerId: 1 })                      // customerId_1
db.customers.createIndex({ customerId: 1 })                   // customerId_1
```

Beispiel: Eine Suchabfrage an einen zusammengesetzten Index anheften

Stellen Sie sich eine Abfrage vor, die nach folgenden `status` Kriterien filtert und sortiert: `orderDate`

```
db.orders.find({ status: "SHIPPED" }).sort({ orderDate: 1 })
```

Der zusammengesetzte Index `status_1_orderDate_1` erfüllt sowohl das Prädikat als auch die Sortierung, sodass keine separate Sortierung erforderlich ist. Wählt der Planer stattdessen`status_1`, muss die Abfrage ihre Ergebnisse sortieren, wodurch eine weitere `SORT` Stufe hinzugefügt wird und mit zunehmender Anzahl übereinstimmender Bestellungen teurer wird. Um den Planer auf den zusammengesetzten Index für diese Form zu beschränken:

```
db.runCommand({planCacheSetFilter: "orders",
```

```
query: { status: "SHIPPED" },
sort: { orderDate: 1 },
indexes: [ "status_1_orderDate_1" ]})
```

Da Werte außerhalb der Form normalisiert werden `{ status: "PENDING" }{ status: "CANCELLED" }`, deckt dieser Filter auch alle anderen Werte von `status` mit derselben Sortierung ab. Er deckt nicht dasselbe Prädikat mit einer anderen Sortierung oder ohne Sortierung ab, bei denen es sich um separate Formen handelt. Bestätigen Sie das Wirksamwerden des Filters mit`explain`:

```
db.orders.find({ status: "SHIPPED" }).sort({ orderDate: 1 }).explain()
```

Die Ausgabe meldet `indexFilterSet: true` und`indexFilterApplied: true`, und die `IXSCAN` Etappennamen`status_1_orderDate_1`.

Beispiel: Ein **`Update an einen selektiven`** Index anheften

Bei einer Aktualisierung müssen die zu ändernden Dokumente gefunden werden, bevor sie geschrieben werden können. Diese Suche verwendet einen Index auf die gleiche Weise wie ein Lesevorgang. Wenn ein Prädikat von mehr als einem Index bereitgestellt werden kann, bestimmt der vom Planer gewählte Index, wie viele Dokumente bei der Aktualisierung untersucht werden.

```
db.orders.updateMany(
  { customerId: 4815, status: "PENDING" },
  { $set: { status: "CANCELLED" } }
)
```

Beides `customerId_1` und `status_1` kann dieses Prädikat erfüllen. Um das Update anzuheften an`customerId_1`:

```
db.runCommand({planCacheSetFilter: "orders",
query: { customerId: 4815, status: "PENDING" },
indexes: [ "customerId_1" ]})
```

Filter für Schreibbefehle erfordern Amazon DocumentDB 5.0.2 oder 8.0.2 und Planner Version 2.0 oder höher. Überprüfen Sie mit einem `explain` der Updates:

```
db.runCommand({explain: {update: "orders",
updates: [{ q: { customerId: 4815, status: "PENDING" },
             u: { $set: { status: "CANCELLED" } },
             multi: true }]}})
```

Der Plan, der den Sieg davonträgt, ist eine UPDATE Stufe höher als IXSCAN die anderecustomerId_1. Derselbe Filter gilt für findAndModify Befehle delete und Befehle, die dieses Abfrage-Shape verwenden, da das Shape den Typ des Schreibens nicht enthält.

Beispiel: Anheften einer Aggregatpipeline und einer **$lookup-Fremdsammlung**

Bei einer Aggregation wird die Form aus der ersten $match Phase und einer $sort unmittelbar darauf folgenden Phase übernommen. Stufen, die darüber hinausgehen, wie z. B. $group oder$project, sind nicht Teil der Form.

```
db.orders.aggregate([
  { $match: { status: "SHIPPED" } },
  { $group: { _id: "$customerId", total: { $sum: "$amount" } } }
])
```

Ein auf das $match Prädikat gesetzter Filter steuert den Index, der zum Lesen orders verwendet wird:

```
db.runCommand({planCacheSetFilter: "orders",
query: { status: "SHIPPED" },
indexes: [ "status_1" ]})
```

Filter gelten auch für die Sammlung, zu der eine $lookup oder $graphLookup -Stufe gehört. Der Filter wird für die fremde Sammlung gesetzt, nicht für die Sammlung, gegen die die Pipeline läuft. In der folgenden Pipeline orders befindet sich die äußere Sammlung und customers ist die ausländische Sammlung:

```
db.orders.aggregate([
  { $lookup: { from: "customers",
               localField: "customerId",
               foreignField: "customerId",
               as: "customer" } }
])
```

Der Join wird customers einmal pro äußerem Dokument geprüft, sodass der dort gewählte Index wiederholt angewendet wird und die damit verbundenen Kosten in der gesamten Pipeline multipliziert werden. Um diesen Scan einem bestimmten Index zuzuordnen, aktivieren Sie den Filter customers für die Gleichheitsform des Join-Schlüssels:

```
db.runCommand({planCacheSetFilter: "customers",
```

```
query: { customerId: 1 },
indexes: [ "customerId_1" ]})
```

Da Werte außerhalb der Form normalisiert werden, ist der 1 obige Wert ein Platzhalter, und jeder Wert erzeugt denselben Filter. Wenn ein Filter für ausländische Sammlungen angewendet wird, werden die oberste Ebene `indexFilterSet` und die `indexFilterApplied` Felder gemeldet, und der Index`true`, für den der Filter ausgewählt wurde, wird auf der Fremdseite des Gewinnerplans angezeigt. Für aktivierte Filter `aggregate` sind Amazon DocumentDB 8.0.2 und Planner Version 3.0 erforderlich.

Verhalten bei Hinweisen und Indexänderungen

- Ein Plan-Cache-Filter hat Vorrang vor einem`hint`. Amazon DocumentDB wendet zuerst den Filter an und wendet dann den Hinweis in der Liste der zulässigen Indizes des Filters an. Wenn der Hinweis einen Index benennt, den der Filter zulässt, wird dieser Index verwendet. Wenn der Hinweis einen Index benennt, den der Filter nicht zulässt, wird der Hinweis ignoriert und der Planer wählt einen der zulässigen Indizes nach den Kosten aus. Beide Hinweisformen verhalten sich gleich, unabhängig davon, ob Sie den Index benennen oder sein Schlüsselmuster angeben.

  Wenn die Sammlung beispielsweise die Indizes`a_1`, `b_1a_1_b_1`, und `foo` hat und Sie einen Filter auf die Form `{query: {a: {$eq: "@"}, b: {$eq: "@"}}, sort: {a: 1}}` mit der Indexliste setzen`["a_1", "a_1_b_1"]`, wird das Ausführen `db.foo.find({ a: 10, b: 20 }).sort({a: 1}).hint({ a: 1 })` verwendet`a_1`, da es im Hinweis benannt ist und vom Filter zugelassen wird.
- Durch das Löschen eines Indexes werden die Filter nicht geändert. Der Filter behält den Indexnamen bei. Obwohl dieser Index fehlt, wird er von der Planung übersprungen. Wenn ein anderer Index in der Liste des Filters die Abfrage verarbeiten kann, gilt der Filter trotzdem. Nur wenn keiner der benannten Indizes verfügbar ist, wird ein `indexFilterApplied` Bericht erstellt`false`. Wenn Sie einen Index mit demselben Namen neu erstellen, ist er wieder gültig.
- Wenn Sie eine Sammlung löschen, werden alle Filter dieser Sammlung entfernt.

Bewährte Methoden für Amazon DocumentDB

Erfahren Sie mehr über bewährte Methoden für die Arbeit mit Amazon DocumentDB (mit MongoDB-Kompatibilität). Dieser Abschnitt wird fortlaufend aktualisiert, wenn neue bewährte Methoden identifiziert werden.

Themen

- Grundlegende Anleitungen für den Betrieb
- Dimensionierung der Instanz
- Arbeiten mit Indizes
- Bewährte Methoden für die Gewährleistung der Sicherheit
- Kostenoptimierung
- Verwendung von Metriken zur Identifizierung von Problemen mit der Leistung
- TTL- und Zeitreihen-Workloads
- Migrationen
- Arbeiten mit Cluster-Parametergruppen
- Abfragen der Aggregationspipeline
- BatchInsert und BatchUpdate

Grundlegende Anleitungen für den Betrieb

Im Folgenden finden Sie grundlegende Betriebsrichtlinien, die jeder bei der Arbeit mit Amazon DocumentDB befolgen sollte. Das Amazon DocumentDB Service Level Agreement verlangt, dass Sie diese Richtlinien befolgen.

- Stellen Sie einen Cluster bereit, der aus zwei oder mehr Amazon DocumentDB-Instances in zwei AWS Availability Zones besteht. Stellen Sie für Produktionsworkloads einen Cluster bereit, der aus drei oder mehr Amazon DocumentDB-Instances in drei Availability Zones besteht.
- Verwenden Sie die Services innerhalb der angegebenen Service Limits. Weitere Informationen finden Sie unter Amazon DocumentDB-Kontingente.
- Überwachen Sie Speicher, CPU, Verbindungen und Speichernutzung. Damit Sie die Systemleistung und Verfügbarkeit aufrechterhalten können, richten Sie Amazon so ein, CloudWatch dass es Sie benachrichtigt, wenn sich die Nutzungsmuster ändern oder wenn Sie sich der Kapazität Ihrer Bereitstellung nähern.
- Skalieren Sie Ihre Instances, wenn Sie die Grenzen der Speicherkapazität beinahe erreicht haben. Ihre Instances sollten mit genügend Rechenressourcen (d. h. RAM, CPU) ausgestattet sein, um Nachfragesteigerungen seitens Ihrer Anwendungen bewältigen zu können.
- Legen Sie die Aufbewahrungszeitraum für Backups so fest, dass sie zu Ihrem Wiederherstellungsziel passt.

- Testen Sie den Failover für Ihren Cluster, um zu verstehen, wie lange der Vorgang für Ihren Anwendungsfall dauert. Weitere Informationen finden Sie unter Amazon DocumentDB-Failover.
- Stellen Sie mit dem Cluster-Endpunkt (siheAmazon DocumentDB-Endpunkte) und im Replikat-Set-Modus (siheVerbindung zu Amazon DocumentDB als Replikatsatz herstellen) eine Verbindung zu Ihrem Amazon DocumentDB-Cluster her, um die Auswirkungen eines Failovers auf Ihre Anwendung zu minimieren.
- Wählen Sie eine Treibereinstellung für die Leseeinstellung aus, die die Leseskalierung maximiert und gleichzeitig die Anforderungen für die Lesekonsistenz Ihrer Anwendung erfüllt. Die Leseeinstellung `secondaryPreferred` ermöglicht Replica-Lesevorgänge, sodass die primäre Instance produktiver sein kann. Weitere Informationen finden Sie unter Präferenzoptionen lesen.
- Entwerfen Sie Ihre Anwendung so, dass Sie im Falle von Netzwerk- und Datenbankfehlern stabil ist. Unterscheiden Sie mithilfe des Fehlermechanismus des Treibers zwischen vorübergehenden Fehlern und persistenten Fehlern. Wiederholen Sie den Vorgang bei vorübergehenden Fehlern mit einem exponentiellen Backoff-Mechanismus (bei Bedarf). Stellen Sie sicher, dass Ihre Anwendung bei der Implementierung von Logik für Wiederholversuche die Datenkonsistenz berücksichtigt.
- Aktivieren Sie den Cluster-Löschschutz für alle Produktions-Cluster oder Cluster mit wertvollen Daten. Bevor Sie einen Amazon DocumentDB-Cluster löschen, erstellen Sie einen letzten Snapshot. Wenn Sie Ressourcen mit bereitstellen CloudFormation, aktivieren Sie den Kündigungsschutz. Weitere Informationen finden Sie unter Kündigungsschutz und Löschschutz.
- Bei der Erstellung eines Amazon DocumentDB-Clusters `--engine-version` ist der ein optionaler Parameter, der standardmäßig die neueste Version der Haupt-Engine verwendet. Die aktuelle Standard-Engine-Version ist 5.0.0 (Hinweis: Amazon DocumentDB 8.0 ist verfügbar, muss aber explizit angegeben werden). Wenn neue Hauptversionen der Engine veröffentlicht werden, `--engine-version` wird die Standard-Engine-Version für aktualisiert, sodass sie die letzte Hauptversion der Engine wiedergibt. Geben Sie daher für Produktions-Workloads und insbesondere für solche, die von Scripting, Automatisierung oder CloudFormation Vorlagen abhängig sind, explizit die gewünschte Hauptversion `--engine-version` an.

Dimensionierung der Instanz

Einer der wichtigsten Aspekte bei der Auswahl einer Instance-Größe in Amazon DocumentDB ist die Größe des Arbeitsspeichers für Ihren Cache. Amazon DocumentDB reserviert ein Drittel des Arbeitsspeichers für seine eigenen Dienste, was bedeutet, dass nur zwei Drittel des Instance-RAM für den Cache verfügbar sind. Daher ist es eine bewährte Methode von Amazon DocumentDB, einen Instance-Typ mit ausreichend RAM zu wählen, um Ihren Arbeitsspeicher (d. h. Daten und

Indizes) im Arbeitsspeicher unterzubringen. Instances mit der richtigen Größe helfen dabei, die Gesamtleistung zu optimieren und potenziell die Kosten zu minimieren I/O . Sie können den Amazon DocumentDB-Größenrechner eines Drittanbieters verwenden, um die Instance-Größe für eine bestimmte Arbeitslast abzuschätzen.

Um festzustellen, ob der Funktionsumfang Ihrer Anwendung in den Arbeitsspeicher passt, überwachen `BufferCacheHitRatio` Sie die Nutzung von Amazon CloudWatch für jede Instance in einem Cluster, die unter Last steht.

Die `BufferCacheHitRatio` CloudWatch Metrik misst den Prozentsatz der Daten und Indizes, die aus dem Speicher-Cache einer Instance bereitgestellt werden (im Vergleich zum Speichervolumen). Im Allgemeinen sollte der Wert von `BufferCacheHitRatio` so hoch wie möglich sein, da das Lesen von Daten aus dem Arbeitssatz-Speicher schneller und kostengünstiger ist als das Lesen vom Speicher-Volume. Obwohl es wünschenswert ist, `BufferCacheHitRatio` möglichst nahe bei 100% zu halten, hängt der beste erreichbare Wert von den Zugriffsmustern und den Leistungsanforderungen Ihrer Anwendung ab. Um das höchstmögliche `BufferCacheHitRatio` beizubehalten, wird empfohlen, dass die Instances in Ihrem Cluster über ausreichend RAM verfügen, damit Ihre Indizes und Arbeitsdatensätze in den Speicher passen.

Wenn Ihre Indizes nicht in den Speicher passen, wird Ihnen ein niedrigeres `BufferCacheHitRatio` angezeigt. Kontinuierliches Lesen von der Festplatte verursacht zusätzliche I/O Kosten und ist nicht besser als das Lesen aus dem Speicher. Wenn Ihr `BufferCacheHitRatio`-Verhältnis niedriger als erwartet ist, skalieren Sie die Instance-Größe für Ihren Cluster, um mehr RAM zur Verfügung zu stellen, damit die Arbeitssatzdaten in den Speicher passen. Wenn das Skalieren der Instance-Klasse zu einem enormen Anstieg im `BufferCacheHitRatio` führt, passte der Arbeitssatz Ihrer Anwendung nicht in den Speicher. Skalieren Sie weiter, bis sich `BufferCacheHitRatio` nach einer Skalierungsoperation nicht mehr drastisch erhöht. Weitere Informationen zur Überwachung von Instance-Metriken finden Sie unter Amazon DocumentDB-Metriken.

Abhängig von Ihren Workload- und Latenzanforderungen kann es akzeptabel sein, dass Ihre Anwendung während der stabilen Zustandsnutzung höhere `BufferCacheHitRatio`-Werte aufweist, das `BufferCacheHitRatio` jedoch periodisch nachlässt, da analytische Abfragen, die eine gesamte Sammlung scannen müssen, auf einer Instance ausgeführt werden. Diese periodischen Einbrüche im `BufferCacheHitRatio` können sich als höhere Latenz für nachfolgende Abfragen manifestieren, die die Arbeitssatzdaten aus dem Speicher-Volume wieder in den Puffercache füllen müssen. Testen Sie Ihre Workloads zunächst in einer Vorproduktionsumgebung mit einer repräsentativen Produktionsauslastung, um sich ein Bild von

den Leistungsmerkmalen zu machen, und **`BufferCacheHitRatio`** bevor Sie den Workload für die Produktion bereitstellen.

Es handelt sich beim `BufferCacheHitRatio` um eine Instance-spezifische Metrik, daher können verschiedene Instances innerhalb desselben Clusters unterschiedliche `BufferCacheHitRatio`-Werte aufweisen, je nachdem, wie Lesevorgänge auf die Primär- und Replikat-Instances verteilt werden. Wenn Ihr betrieblicher Workload nicht mit periodischen Erhöhungen der Latenz durch erneutes Auffüllen des Arbeitssatzcache nach dem Ausführen analytischer Abfragen umgehen kann, sollten Sie versuchen, den Puffercache des regulären Workloads von dem der analytischen Abfragen zu isolieren. Sie können eine vollständige `BufferCacheHitRatio`-Isolation erreichen, indem Sie betriebliche Abfragen an die Primär-Instance und analytische Abfragen nur an die Replikat-Instances weiterleiten. Sie können auch eine partielle Isolation erreichen, indem Sie analytische Abfragen an eine bestimmte Replikat-Instance weiterleiten, mit dem Verständnis, dass ein gewisser Prozentsatz der regulären Abfragen auch auf diesem Replikat ausgeführt wird und möglicherweise betroffen sein könnte.

Angemessene `BufferCacheHitRatio`-Werte hängen von Ihrem Anwendungsfall und Ihren Anwendungsanforderungen ab. Es gibt keinen besten oder minimalen Wert für diese Metrik. Nur Sie können entscheiden, ob der Kompromiss von einem vorübergehend niedrigeren `BufferCacheHitRatio` aus Kosten- und Leistungsperspektive akzeptabel ist.

Arbeiten mit Indizes

Indizes erstellen

Wenn Sie Daten in Amazon DocumentDB importieren, sollten Sie Ihre Indizes erstellen, bevor Sie große Datensätze importieren. Sie können das Amazon DocumentDB Index Tool verwenden, um Indizes aus einer laufenden MongoDB-Instance oder einem `mongodump` Verzeichnis zu extrahieren und diese Indizes in einem Amazon DocumentDB-Cluster zu erstellen. Weitere Hinweise zu Migrationen finden Sie unter Migration und Aktualisierung von Amazon DocumentDB.

Index-Selektivität

Beschränken Sie die Indexerstellung auf Felder, bei denen die Anzahl der doppelten Werte weniger als 1% der Gesamtzahl der Dokumente in der Sammlung beträgt. Wenn Ihre Sammlung beispielsweise 100.000 Dokumente enthält, erstellen Sie Indizes nur für Felder, in denen derselbe Wert 1.000 Mal oder weniger vorkommt.

Die Wahl eines Index mit einer hohen Anzahl von eindeutigen Werten (d. h. einer hohen Kardinalität) gewährleistet, dass Filteroperationen eine geringe Anzahl von Dokumenten zurückgeben, wodurch eine gute Leistung während der Index-Scans erzielt wird. Ein Beispiel für einen Index hoher Kardinalität ist ein einzigartiger Index, der garantiert, dass Gleichheitsprädikate höchstens ein einziges Dokument zurückgeben. Beispiele für niedrige Kardinalität sind ein Index über ein boolesches Feld und ein Index über einen Wochentag. Aufgrund ihrer schlechten Leistung ist es unwahrscheinlich, dass der Abfragenoptimierer der Datenbank Indizes mit niedriger Kardinalität auswählt. Gleichzeitig verbrauchen Indizes mit niedriger Kardinalität weiterhin Ressourcen wie Speicherplatz und. I/Os Als Faustregel gilt, dass Sie Indizes für Felder verwenden sollten, bei denen die typische Wertehäufigkeit 1 % der Gesamtsammlungsgröße oder weniger beträgt.

Darüber hinaus wird empfohlen, nur Indizes für Felder zu erstellen, die häufig als Filter verwendet werden, und regelmäßig nach nicht verwendeten Indizes zu suchen. Weitere Informationen finden Sie unter Wie analysiere ich die Indexnutzung und identifiziere ungenutzte Indizes?.

Auswirkung von Indizes auf das Schreiben von Daten

Zwar können Indizes die Abfrageleistung verbessern, da nicht jedes Dokument in einer Sammlung gescannt werden muss, diese Verbesserung erfordert jedoch einen Kompromiss. Für jeden Index einer Sammlung muss die Datenbank jedes Mal, wenn ein Dokument eingefügt, aktualisiert oder gelöscht wird, die Sammlung aktualisieren und die Felder in jeden der Indizes für die Sammlung schreiben. Wenn eine Sammlung beispielsweise über neun Indizes verfügt, muss die Datenbank zehn Schreibvorgänge durchführen, bevor die Operation dem Kunden bestätigt wird. Somit führt jeder zusätzliche Index zu einer zusätzlichen Schreiblatenz und einer Erhöhung des insgesamt genutzten I/O Speichers.

Cluster-Instances müssen über eine angemessene Größe verfügen, um den gesamten Arbeitsspeicher zu erhalten. Dadurch entfällt die Notwendigkeit, ständig Indexseiten vom Speichervolumen zu lesen, was sich negativ auf die Leistung auswirkt und höhere I/O Kosten verursacht. Weitere Informationen finden Sie unter Dimensionierung der Instanz.

Um eine optimale Leistung zu erzielen, sollten Sie die Anzahl der Indizes in Ihren Sammlungen minimieren und nur die Indizes hinzufügen, die zur Verbesserung der Leistung bei häufigen Abfragen erforderlich sind. Bei variierenden Workloads gilt als gute Richtschnur, die Anzahl der Indizes pro Sammlung auf fünf oder weniger zu beschränken.

Identifizierung fehlender Indizes

Es hat sich bewährt, fehlende Indizes regelmäßig zu identifizieren. Weitere Informationen finden Sie unter Wie identifiziere ich fehlende Indizes?.

Identifizieren ungenutzter Indizes

Es hat sich bewährt, ungenutzte Indizes regelmäßig zu identifizieren und zu entfernen. Weitere Informationen finden Sie unter Wie analysiere ich die Indexnutzung und identifiziere ungenutzte Indizes?.

Bewährte Methoden für die Gewährleistung der Sicherheit

Bewährte Sicherheitsmethoden für Amazon DocumentDB finden Sie unter. Bewährte Sicherheitsmethoden für Amazon DocumentDB

Kostenoptimierung

Die folgenden bewährten Methoden können Ihnen helfen, Ihre Kosten bei der Verwendung von Amazon DocumentDB zu verwalten und zu minimieren. Preisinformationen finden Sie unter Preise für Amazon DocumentDB (mit MongoDB-Kompatibilität) und häufig gestellte Fragen zu Amazon DocumentDB (mit MongoDB-Kompatibilität).

- Erstellen Sie Gebührenlimit-Warnungen für Schwellenwerte von 50 Prozent und 75 Prozent Ihrer erwarteten Rechnung für den Monat. Weitere Informationen zum Erstellen von Gebührenlimit-Warnungen finden Sie unter Erstellen einer Gebührenlimit-Warnung.
- Die Architektur von Amazon DocumentDB trennt Speicher und Rechenleistung, sodass selbst ein Single-Instance-Cluster äußerst robust ist. Das Cluster-Speicher-Volume repliziert Daten auf sechs Arten in drei Availability Zones und bietet unabhängig von der Anzahl der Instances im Cluster eine extrem hohe Beständigkeit. Ein typischer Produktions-Cluster verfügt über mindestens drei Instances, um eine hohe Verfügbarkeit bereitzustellen. Sie können jedoch die Kosten optimieren, indem Sie einen einzelnen Instance-Entwicklungs-Cluster verwenden, wenn keine hohe Verfügbarkeit erforderlich ist.
- Halten Sie bei Entwicklungs- und Testszenarien einen Cluster an, wenn er nicht mehr benötigt wird, und starten Sie den Cluster, wenn die Entwicklung fortgesetzt wird. Weitere Informationen finden Sie unter Stoppen und Starten eines Amazon DocumentDB-Clusters.

- Sowohl TTL- als auch Change-Streams entstehen I/O, wenn Daten geschrieben, gelesen und gelöscht werden. Wenn Sie diese Funktionen aktiviert haben, sie aber in Ihrer Anwendung nicht nutzen, kann die Deaktivierung der Funktionen zur Kostensenkung beitragen.

Verwendung von Metriken zur Identifizierung von Problemen mit der Leistung

Themen

- Anzeigen von Leistungsmetriken
- Einen CloudWatch Alarm einstellen
- Auswerten von Leistungsmetriken
- Bewertung der Nutzung der Amazon DocumentDB-Instance anhand von Metriken CloudWatch
- Optimieren von Abfragen

Um Leistungsprobleme zu identifizieren, die durch unzureichende Ressourcen und andere häufig auftretende Engpässe verursacht werden, können Sie die für Ihren Amazon DocumentDB-Cluster verfügbaren Metriken überwachen.

Anzeigen von Leistungsmetriken

Überwachen Sie die Leistungsmetriken regelmäßig, um die Durchschnitts-, Höchst- und Mindestwerte für verschiedene Zeitbereiche anzuzeigen. Auf diese Weise können Sie feststellen, wenn die Leistung nachlässt. Sie können CloudWatch Amazon-Alarme auch für bestimmte Metrik-Schwellenwerte einrichten, sodass Sie benachrichtigt werden, wenn diese erreicht werden.

Um Probleme mit der Leistung zu beheben, müssen Sie die Basisleistung des Systems kennen. Nachdem Sie einen neuen Cluster eingerichtet haben und er mit einer typischen Workload ausgeführt wird, erfassen Sie die Durchschnitts-, Höchst- und Mindestwerte aller Leistungsmetriken in unterschiedlichen Intervallen (z. B. 1 Stunde, 24 Stunden, 1 Woche, 2 Wochen). Auf diese Weise erhalten Sie eine Vorstellung davon, was normal ist. Dies hilft, um Vergleichswerte für Betriebsstunden während und außerhalb von Spitzenbelastungen zu erhalten. Sie können diese Informationen anschließend verwenden, um festzustellen, wann die Leistung unter Standardwerte absinkt.

Sie können Leistungskennzahlen mithilfe von oder anzeigen. AWS-Managementkonsole AWS CLI Weitere Informationen finden Sie unter CloudWatch Daten anzeigen.

Einen CloudWatch Alarm einstellen

Informationen zum Einstellen eines CloudWatch Alarms finden Sie unter Verwenden von CloudWatch Amazon-Alarmen im CloudWatch Amazon-Benutzerhandbuch.

Auswerten von Leistungsmetriken

Eine Instance besitzt mehrere unterschiedliche Kategorien von Metriken. Die Art und Weise, wie Sie bestimmen, welche Werte akzeptabel sind, ist von der Metrik abhängig.

CPU

- CPU-Auslastung — Der Prozentsatz der verwendeten Computerverarbeitungskapazität.

Arbeitsspeicher

- Freigegebener Speicher — Wie viel RAM auf der Instance verfügbar ist.
- Swap-Nutzung — Wie viel Swap-Speicherplatz wird von der Instance verwendet, in Megabyte.

Input/output Operationen

- IOPS lesen, IOPS schreiben — Die durchschnittliche Anzahl von Lese- oder Schreibvorgängen auf der Festplatte pro Sekunde.
- Leselatenz, Schreiblatenz — Die durchschnittliche Zeit für einen Lese- oder Schreibvorgang in Millisekunden.
- Lesedurchsatz, Schreibdurchsatz — Die durchschnittliche Anzahl von Megabyte, die pro Sekunde von der Festplatte gelesen oder auf die Festplatte geschrieben werden.
- Datenträgerwarteschlangentiefe — Die Anzahl der I/O Vorgänge, die darauf warten, auf die Festplatte geschrieben oder von der Festplatte gelesen zu werden.

Netzwerkdatenverkehr

- Netzwerkempfangsdurchsatz, Netzwerkübertragungsdurchsatz — Die Rate des Netzwerkverkehrs zur und von der Instance in Megabyte pro Sekunde.

Datenbankverbindungen

- DB-Verbindungen — Die Anzahl der Client-Sitzungen, die mit der Instance verbunden sind.

Allgemein ausgedrückt, sind die zulässigen Werte für Leistungsmetriken davon abhängig, wie die Basisleistung aussieht und welche Aufgaben von Ihrer Anwendung ausgeführt werden. Prüfen Sie, ob dauerhafte oder tendenzielle Abweichungen von Ihrer Ausgangsbasis vorliegen.

Nachstehend folgen Empfehlungen und Ratschläge zu bestimmten Arten von Metriken:

- Hoher CPU-Verbrauch — Hohe Werte für die CPU-Auslastung könnten angemessen sein, vorausgesetzt, sie entsprechen Ihren Zielen für Ihre Anwendung (wie Durchsatz oder Parallelität) und werden erwartet. Wenn die CPU-Auslastung konsistent mehr als 80 Prozent beträgt, sollten Sie eine Aufwärtsskalierung Ihrer Instances in Betracht ziehen.
- Hoher RAM-Verbrauch — Wenn Ihre `FreeableMemory` Metrik häufig unter 10% des gesamten Instance-Speichers fällt, sollten Sie eine Skalierung Ihrer Instances in Betracht ziehen. Weitere Informationen darüber, was passiert, wenn Ihre DocumentDB-Instance einem hohen Speicherverbrauch ausgesetzt ist, finden Sie unter Amazon DocumentDB Resource Governance.
- Swap-Nutzung — Diese Metrik sollte bei oder nahe Null bleiben. Wenn die Swap-Nutzung erheblich ist, sollten Sie eine Aufwärtsskalierung Ihrer Instances in Betracht ziehen.
- Netzwerkverkehr — Sprechen Sie mit Ihrem Systemadministrator, um zu erfahren, wie hoch der erwartete Durchsatz für Ihr Domänennetzwerk und Ihre Internetverbindung ist. Überprüfen Sie den Netzwerkdatenverkehr, wenn der Durchsatz dauerhaft unter dem erwarteten Wert liegt.
- Datenbankverbindungen — Erwägen Sie, Datenbankverbindungen einzuschränken, wenn Sie eine hohe Anzahl von Benutzerverbindungen sowie eine Verringerung der Instance-Leistung und Reaktionszeit feststellen. Die optimale Anzahl der Benutzerverbindungen für Ihre Instance ist von der Instance-Klasse und der Komplexität der ausgeführten Operationen abhängig. Bei Problemen mit Leistungsmetriken sollten Sie zunächst die am häufigsten verwendeten und kostspieligsten Abfragen anpassen, um festzustellen, ob dies den Druck auf die Systemressourcen verringert und die Leistung verbessert.

Wenn Ihre Abfragen optimiert sind und ein Problem weiterhin besteht, sollten Sie erwägen, Ihre Amazon DocumentDB-Instance-Klasse auf eine Klasse mit mehr Ressourcen (CPU, RAM, Festplattenspeicher, Netzwerkbandbreite, I/O Kapazität) zu aktualisieren, die mit dem aufgetretenen Problem zusammenhängt.

Bewertung der Nutzung der Amazon DocumentDB-Instance anhand von Metriken CloudWatch

Sie können CloudWatch Metriken verwenden, um Ihren Instance-Durchsatz zu beobachten und festzustellen, ob Ihre Instance-Klasse ausreichend Ressourcen für Ihre Anwendungen bereitstellt.

Informationen zu Ihren Instance-Klassenkontingenten finden Sie in den Spezifikationen für Ihre Instance-Klasse, um Ihre Netzwerkleistung zu ermitteln. Kontingente für Instanzen

Wenn sich Ihre Instance-Nutzung dem Instance-Klassenlimit nähert, verlangsamt sich möglicherweise die Leistung. Die CloudWatch Metriken können diese Situation bestätigen, sodass Sie eine manuelle Skalierung auf eine größere Instance-Klasse planen können.

Kombinieren Sie die folgenden CloudWatch Metrikwerte, um herauszufinden, ob Sie sich dem Instance-Klassenlimit nähern:

- NetworkThroughput— Der Umfang des Netzwerkdurchsatzes, der von den Clients für jede Instance im Amazon DocumentDB-Cluster empfangen und übertragen wurde. Dieser Durchsatzwert beinhaltet nicht den Netzwerkverkehr zwischen Instances im Cluster und dem Cluster-Speichervolume.
- StorageNetworkThroughput— Der Umfang des Netzwerkdurchsatzes, der von jeder Instance im Amazon DocumentDB-Cluster empfangen und an das Amazon DocumentDB-Cluster-Speichervolume gesendet wurde.

Fügen Sie den hinzu NetworkThroughput, um den Netzwerkdurchsatz StorageNetworkThroughput zu ermitteln, der von jeder Instance in Ihrem Amazon DocumentDB-Cluster vom Amazon DocumentDB-Cluster empfangen und an das Amazon DocumentDB-Cluster-Speicher-Volume gesendet wurde. Das Instance-Klassenlimit für Ihre Instance sollte größer sein als die Summe dieser beiden kombinierten Metriken.

Sie können die folgenden Metriken verwenden, um zusätzliche Details des Netzwerkverkehrs Ihrer Client-Anwendungen beim Senden und Empfangen zu überprüfen:

- NetworkReceiveThroughput— Der Umfang des Netzwerkdurchsatzes, den jede Instance im Amazon DocumentDB-Cluster von den Clients erhält. Dieser Durchsatz beinhaltet nicht den Netzwerkverkehr zwischen Instances im Cluster und dem Cluster-Speichervolume.
- NetworkTransmitThroughput— Der Umfang des Netzwerkdurchsatzes, der von jeder Instance im Amazon DocumentDB-Cluster an die Clients gesendet wird. Dieser Durchsatz beinhaltet nicht den Netzwerkverkehr zwischen Instances im Cluster und dem Cluster-Speichervolume.
- StorageNetworkReceiveThroughput— Der Umfang des Netzwerkdurchsatzes, der von jeder Instance im Cluster vom Amazon DocumentDB-Cluster-Speichervolume empfangen wurde.
- StorageNetworkTransmitThroughput— Der Umfang des Netzwerkdurchsatzes, der von jeder Instance im Cluster an das Amazon DocumentDB-Cluster-Speichervolume gesendet wurde.

Addieren Sie all diese Metriken, um zu bewerten, wie Ihre Netzwerknutzung im Vergleich zum Instance-Klassenlimit abschneidet. Das Instance-Klassenlimit sollte größer sein als die Summe dieser kombinierten Metriken.

Die Netzwerklimits und die CPU-Auslastung durch eine Instance beruhen auf Gegenseitigkeit. Wenn der Netzwerkdurchsatz steigt, steigt auch die CPU-Auslastung. Die Überwachung der CPU- und Netzwerknutzung gibt Aufschluss darüber, wie und warum die Ressourcen erschöpft werden.

Sie können Folgendes in Betracht ziehen, um die Netzwerknutzung zu minimieren:

- Verwenden einer größeren Instance-Klasse
- Aufteilen der Schreibanforderungen in Batches, um die Gesamttransaktionen zu reduzieren
- Weiterleiten der schreibgeschützten Workload an eine schreibgeschützte Instance
- Löschen aller ungenutzten Indizes

Optimieren von Abfragen

Eine der besten Möglichkeiten zur Verbesserung der Cluster-Leistung besteht darin, die am häufigsten verwendeten und ressourcenintensivsten Abfragen so anzupassen, dass ihre Ausführung weniger aufwändig wird.

Mit dem Profiler (siehe Profilierung von Amazon DocumentDB-Vorgängen) können Sie die Ausführungszeit und Details der Vorgänge protokollieren, die für Ihren Cluster ausgeführt wurden. Profiler ist nützlich für die Überwachung der langsamsten Operationen in Ihrem Cluster, um die Leistung einzelner Abfragen und die allgemeine Cluster-Leistung zu verbessern.

Sie können den Befehl `explain` auch verwenden, um zu erfahren, wie ein Abfrageplan für eine bestimmte Abfrage analysiert wird. Mithilfe dieser Informationen können Sie eine Abfrage oder zugrundeliegende Sammlung ändern, um die Abfrageleistung zu verbessern (z. B. Hinzufügen eines Indexes).

TTL- und Zeitreihen-Workloads

Das Löschen von Dokumenten, das aus dem Ablauf des TTL-Index resultiert, ist ein hochaufwendiges Verfahren. Es wird nicht garantiert, dass Dokumente innerhalb eines bestimmten Zeitraums gelöscht werden. Faktoren wie die Größe der Instance, die Ressourcenauslastung der Instance, die Dokumentgröße, der Gesamtdurchsatz, die Anzahl der Indizes und die Frage, ob die Indizes und der Arbeitssatz in den Speicher passen, können den Zeitpunkt beeinflussen, zu dem abgelaufene Dokumente durch den TTL-Prozess gelöscht werden.

Wenn der TTL-Monitor Ihre Dokumente löscht, verursacht jedes Löschen I/O Kosten, was Ihre Rechnung erhöht. Wenn der Durchsatz und die TTL-Löschraten steigen, sollten Sie aufgrund der erhöhten Nutzung mit einer höheren Rechnung rechnen. I/O Wenn Sie jedoch keinen TTL-Index zum Löschen von Dokumenten erstellen, sondern Dokumente stattdessen nach Zeit in Sammlungen unterteilen und diese Sammlungen einfach löschen, wenn sie nicht mehr benötigt werden, fallen keine I/O-Kosten an. Dies kann erheblich kostengünstiger sein als die Verwendung eines TTL-Indexes.

Bei Zeitreihen-Workloads können Sie erwägen, fortlaufende Sammlungen anstelle eines TTL-Indexes zu erstellen, da rollierende Sammlungen eine bessere Methode zum Löschen von Daten sein können und weniger intensiv sind. I/O Wenn Sie große Sammlungen haben (insbesondere Sammlungen über 1 TB) oder wenn die I/O TTL-Löschkosten ein Problem darstellen, partitionieren Sie Dokumente nach Zeit in Sammlungen und löschen Sie Sammlungen, wenn die Dokumente nicht mehr benötigt werden. Sie können eine Sammlung pro Tag oder eine Sammlung pro Woche erstellen, abhängig von der Datenaufnahmerate. Bei je nach Anwendung variierenden Anforderungen gilt als gute Faustregel, besser über mehr kleinere Sammlungen als über einige große Sammlungen zu verfügen. Das Löschen dieser Sammlungen verursacht keine I/O Kosten und kann schneller und kostengünstiger sein als die Verwendung eines TTL-Index.

Migrationen

Als bewährte Methode für die Migration von Daten zu Amazon DocumentDB sollten Sie zunächst Ihre Indizes in Amazon DocumentDB erstellen und dann Ihre Daten migrieren. Indizes zuerst zu erstellen, kann die Gesamtzeit verkürzen und die Geschwindigkeit der Migration erhöhen. Dazu können Sie das Amazon DocumentDB Index Tool verwenden. https://github.com/awslabs/amazon-documentdb-tools Weitere Informationen zu Migrationen finden Sie unter. Migration und Aktualisierung von Amazon DocumentDB

Wir empfehlen außerdem, Ihre Anwendung vor der Migration Ihrer Produktionsdatenbank vollständig auf Amazon DocumentDB zu testen und dabei Funktionalität, Leistung, Betrieb und Kosten zu berücksichtigen.

Arbeiten mit Cluster-Parametergruppen

Testen Sie Änderungen an Cluster-Parametergruppen auf einem Testcluster, bevor Sie die Änderungen auf Ihre Produktionscluster anwenden. Weitere Informationen zum Sichern Ihres Clusters finden Sie unter Sichern und Wiederherstellen in Amazon DocumentDB.

Abfragen der Aggregationspipeline

Wenn Sie eine Aggregation-Pipeline-Abfrage mit mehreren Stufen erstellen und nur eine Teilmenge der Daten in der Abfrage auswerten, verwenden Sie die `$match`-Stufe als erste Stufe oder am Anfang der Pipeline. Wenn Sie `$match` zuerst verwenden, wird die Anzahl der Dokumente reduziert, die nachfolgende Stufen innerhalb der Aggregation-Pipeline-Abfrage verarbeiten müssen, wodurch die Leistung Ihrer Abfrage verbessert wird.

`BatchInsert und BatchUpdate`

Wenn Sie eine hohe Anzahl gleichzeitiger `batchInsert` and/or `batchUpdate` Operationen ausführen und die Menge von `FreeableMemory` (CloudWatch Metric) auf Ihrer primären Instance auf Null geht, können Sie entweder die Parallelität des Batch-Insert- oder Aktualisierungs-Workloads reduzieren oder, falls die Parallelität des Workloads nicht reduziert werden kann, die Instance-Größe erhöhen, um den Umfang von zu erhöhen. `FreeableMemory`

Amazon DocumentDB-Indizes verwalten

Themen

- Erstellung eines Amazon DocumentDB-Indexes
- Pflege der Amazon DocumentDB-Indizes

Erstellung eines Amazon DocumentDB-Indexes

Für die Erstellung von Indizes in Amazon DocumentDB müssen eine Reihe von Entscheidungen getroffen werden:

- Wie schnell muss es abgeschlossen werden?
- Kann während des Builds auf die Sammlung nicht zugegriffen werden?
- Wie viel Rechenleistung einer Instanz kann dem Build zugewiesen werden?
- Welcher Indextyp sollte erstellt werden?

Dieser Abschnitt hilft Ihnen bei der Beantwortung dieser Fragen und enthält die Befehle und Überwachungsbeispiele, um einen Amazon DocumentDB-Index für Ihre instanzbasierte Clustersammlung zu erstellen.

Richtlinien

Die folgenden Richtlinien beinhalten grundlegende Grenzwerte und Kompromisse bei der Konfiguration bei der Erstellung neuer Indizes:

- Unterstützung für Amazon DocumentDB-Versionen — Während die Single-Worker-Indizierung in allen Amazon DocumentDB-Versionen unterstützt wird, wird die Indexierung mehrerer Worker nur in den Amazon DocumentDB-Versionen 4.0, 5.0 und 8.0 unterstützt.
- Leistungseinbußen — Eine Erhöhung der Anzahl der Worker im Indexerstellungsprozess erhöht die CPU-Auslastung und die Lese-I/O auf der primären Instance Ihrer Amazon DocumentDB-Datenbank. Die Ressourcen, die für die Erstellung eines neuen Indexes benötigt werden, stehen Ihrem laufenden Workload nicht zur Verfügung.
- Elastische Cluster — Parallele Indizierung wird auf elastischen Amazon DocumentDB-Clustern nicht unterstützt.
- Serverlose Cluster — Für die parallele Indizierung auf serverlosen Clustern ist eine Mindestkapazität von mindestens 32 DCUs erforderlich. Unterhalb dieses Schwellenwerts wird der `workers` Parameter akzeptiert, aber der Indexaufbau wird mit einem einzigen Worker ausgeführt. Die maximale Anzahl paralleler Worker skaliert mit Ihrer konfigurierten Mindestkapazität: Ungefähr `min_DCU / 8` Worker sind verfügbar. Beispielsweise können Sie bei 32 DCUs bis zu 4 Worker einsetzen, und bei 64 DCUs können Sie bis zu 8 Worker einsetzen.

  > **Note**
  >
  > Nach der Änderung der Mindestkapazitätseinstellung ist ein Cluster-Neustart erforderlich, damit das neue Limit für parallele Worker wirksam wird.

- Maximale Anzahl an Workern (instanzbasiert) — Die maximale Anzahl von Workern, die Sie konfigurieren können, hängt von der Größe Ihrer primären Instance in Ihrem Datenbank-Cluster ab. Dies ist die Hälfte der Gesamtzahl der vCPUs auf der primären Instance Ihres Datenbank-Clusters. Beispielsweise können Sie maximal 32 Worker auf einer db.r6g.16xlarge-Instance mit 64 vCPUs ausführen.

  > **Note**
  >
  > Parallele Worker werden nur auf Instanzen ab 2xlarge unterstützt.

- Mindestanzahl an Workern — Die Mindestanzahl von Workern, die Sie konfigurieren können, ist einer. Die Standardeinstellung für die Indexerstellung auf instanzbasierten Clustern ist zwei Worker. Sie können die Anzahl der Worker jedoch auf einen reduzieren, indem Sie die Option „Worker-Threads“ verwenden. Dadurch wird der Prozess mit einem einzigen Worker ausgeführt.
- Indexkomprimierung — Amazon DocumentDB unterstützt keine Indexkomprimierung. Die Datengrößen für Indizes sind möglicherweise größer als bei Verwendung anderer Optionen.
- Indizierung mehrerer Sammlungen — Die Hälfte der vCPUs auf der primären Instanz Ihres Datenbank-Clusters kann für konfigurierte Worker verwendet werden, die Indexerstellung für mehrere Sammlungen durchführen.
- Indextypen — Eine vollständige Erläuterung der unterstützten Indextypen in Amazon DocumentDB finden Sie im Datenbank-Blog unter So indexieren Sie auf Amazon DocumentDB. AWS

Erste Schritte

Verwenden Sie den Befehl, um mit der Indexerstellung für eine Sammlung zu beginnen. `createIndexes` Standardmäßig führt der Befehl zwei parallele Worker aus, wodurch die Geschwindigkeit der Indexerstellung um das Zweifache erhöht wird.

Der folgende Befehlsvorgang veranschaulicht beispielsweise, wie ein Index für das Feld „user_name“ in einem Dokument erstellt und die Geschwindigkeit des Indizierungsprozesses auf vier Worker erhöht wird:

1. Erstellen Sie Indizes mithilfe von zwei parallelen Workern im Cluster:

```
db.runCommand({"createIndexes":"test","indexes":[{"key": {"user_name":1},
 "name":"username_idx"}]})
```

2. Um die Geschwindigkeit der Indexerstellung zu optimieren, können Sie die Anzahl der Worker mithilfe der Option „Worker-Threads“ (`"workers":<number>`) im `db.runCommand` `createIndexes` Befehl angeben.

   Erhöhen Sie die Geschwindigkeit des Prozesses auf vier parallele Worker:

```
db.runCommand({"createIndexes":"test","indexes":[{"key": {"user_name":1},
 "name":"username_idx", "workers":4}]})
```

> **Note**
>
> Je höher die Anzahl der Arbeiter ist, desto schneller schreitet die Indexerstellung voran. Je höher jedoch die Anzahl der Worker steigt, desto höher steigt die Last auf den vCPUs und Read I/O Ihrer primären Instance. Stellen Sie sicher, dass Ihr Cluster ausreichend bereitgestellt ist, um die erhöhte Belastung zu bewältigen, ohne andere Workloads zu beeinträchtigen.

Fortschrittsstatus bei der Indizierung

Bei der Indexerstellung werden Sammlungen initialisiert, gescannt, Schlüssel sortiert und schließlich Schlüssel mithilfe eines Indexgenerators eingefügt. Der Prozess umfasst bis zu sechs Stufen, wenn Sie ihn im Vordergrund ausführen, und bis zu neun Stufen, wenn Sie ihn im Hintergrund ausführen. Sie können Statuskennzahlen wie den Prozentsatz der Fertigstellung, die Gesamtzahl der gescannten Speicherblöcke, sortierte Schlüssel und eingefügte Schlüssel schrittweise anzeigen.

Überwachen Sie den Fortschritt des Indizierungsprozesses, indem Sie den `db.currentOp()` Befehl in der Mongo-Shell verwenden. Ein hundertprozentiger Abschluss der letzten Phase zeigt, dass alle Indizes erfolgreich erstellt wurden:

```
db.currentOp({"command.createIndexes": { $exists : true } })
```

> **Note**
>
> Das Anzeigen des Fortschritts der Indizierung wird nur in Amazon DocumentDB 5.0 unterstützt.

Typen zum Erstellen von Indizes

Die vier Typen von Indexerstellungen sind:

- Vordergrund — Der Vordergrundindex-Build blockiert alle anderen Datenbankoperationen, bis der Index erstellt wird. Der Amazon DocumentDB-Vordergrund-Build besteht aus fünf Stufen.
- Foreground (einzigartig) — Vordergrundindex-Builds für ein einzelnes Dokument (eindeutig) blockieren andere Datenbankoperationen wie reguläre Vordergrund-Builds. Im Gegensatz

zum grundlegenden Vordergrund-Build verwendet der einzigartige Build eine zusätzliche Stufe (Sortierung von Schlüsseln 2), um nach doppelten Schlüsseln zu suchen. Der (einzigartige) Vordergrund-Build besteht aus sechs Stufen.

- Hintergrund — Bei der Indexerstellung im Hintergrund können andere Datenbankoperationen im Vordergrund ausgeführt werden, während der Index erstellt wird. Der Amazon DocumentDB-Hintergrund-Build besteht aus acht Stufen.
- Hintergrund (einzigartig) — Index-Builds im Hintergrund für ein einzelnes Dokument (eindeutig) ermöglichen die Ausführung anderer Datenbankoperationen im Vordergrund, während der Index erstellt wird. Im Gegensatz zur grundlegenden Erstellung im Hintergrund verwendet der eindeutige Build eine zusätzliche Phase (Sortierung der Schlüssel 2), um nach doppelten Schlüsseln zu suchen. Der (einzigartige) Hintergrund-Build besteht aus neun Stufen.

Phasen der Indexerstellung

Stage	Vordergrund	Vordergrund (einzigartig)	Hintergrund	Hintergrund (einzigartig)
Initialisieren	1	1	1	1
Index erstellen: initialisieren	2	2	2	2
Gebäudeindex: Sammlung wird gescannt	3	3	3	3
Gebäudeindex: Schlüssel 1 sortieren	4	4	4	4
Gebäudeindex: Schlüssel 2 sortieren		5		5
Gebäudeindex: Schlüssel einfügen	5	6	5	6

Stage	Vordergrund	Vordergrund (einzigartig)	Hintergrund	Hintergrund (einzigartig)
validieren: Index scannen			6	7
validieren: Tupel sortieren			7	8
validieren: Sammlung scannen			8	9

- initialisieren — createIndex bereitet den Index-Builder vor. Diese Phase sollte sehr kurz sein.
- Index erstellen: Initialisieren — Der Indexgenerator bereitet sich auf die Erstellung des Indexes vor. Diese Phase sollte sehr kurz sein.
- Index erstellen: Sammlung scannen — Der Indexgenerator führt einen Sammlungsscan durch, um Indexschlüssel zu sammeln. Die Maßeinheit ist „Blöcke“.

> **Note**
>
> Wenn mehr als ein Worker für die Indexerstellung konfiguriert ist, wird dies in dieser Phase angezeigt. Die Phase „Erfassung scannen“ ist die einzige Phase, in der während der Indexerstellung mehrere Worker verwendet werden. In allen anderen Phasen wird ein einzelner Worker angezeigt.

- Index erstellen: Schlüssel sortieren 1 — Der Indexgenerator sortiert die gesammelten Indexschlüssel. Die Maßeinheit ist „Schlüssel“.
- Index erstellen: Schlüssel sortieren 2 — Der Indexgenerator sortiert die gesammelten Indexschlüssel, die toten Tupeln entsprechen. Diese Phase existiert nur für die Erstellung eines eindeutigen Indexes. Die Maßeinheit ist „Schlüssel“.
- Index erstellen: Schlüssel einfügen — Der Indexgenerator fügt Indexschlüssel in den neuen Index ein. Die Maßeinheit ist „Schlüssel“.
- validieren: Index scannen — createIndex durchsucht den Index, um Schlüssel zu finden, die überprüft werden müssen. Die Maßeinheit ist „Blöcke“.
- Validieren: Sortieren von Tupeln — CreateIndex sortiert die Ausgabe der Index-Scan-Phase.

- Validieren: Sammlung wird gescannt — CreateIndex durchsucht die Sammlung, um die Indexschlüssel zu überprüfen, die in den beiden vorherigen Phasen gefunden wurden. Die Maßeinheit ist „Blöcke“.

Beispiel für die Ausgabe der Indexerstellung

Im folgenden Ausgabebeispiel (Erstellung des Vordergrundindexes) wird der Status der Indexerstellung angezeigt. Das Feld „msg“ fasst den Fortschritt des Builds zusammen, indem es die Phase und den Prozentsatz der Fertigstellung des Builds angibt. Das Feld „Workers“ gibt die Anzahl der Worker an, die in dieser Phase der Indexerstellung verwendet wurden. Im Feld „Fortschritt“ werden die tatsächlichen Zahlen angezeigt, anhand derer der Prozentsatz der Fertigstellung berechnet wurde.

> **Note**
>
> Die Felder „currentIndexBuildName“, „msg“ und „progress“ werden in Amazon DocumentDB Version 4.0 nicht unterstützt.

```
{
    "inprog" : [{
    …
        "command": {
            "createIndexes": "test",
            "indexes": [{
                "v": 2,
                "key": {
                    "user_name": 1
                },
                "name": "user_name_1"
            }],
            "lsid": {
                "id": UUID(“094d0fba-8f41-4373-82c3-7c4c7b5ff13b”)
            },
            "$db": "test"
        },
        "currentIndexBuildName": user_name_1,
        "msg": "Index Build: building index number_1, stage 6/6 building index:
 656860/1003520 (keys) 65%",
        "workers": 1,
```

```
        "progress": {
            "done": 656861,
            "total": 1003520
        },
    …
    ],

    "ok" : 1
}
```

Pflege der Amazon DocumentDB-Indizes

Themen

- Indexblähung
- Indexverwaltung mithilfe von Reindex

Indexblähung

Amazon DocumentDB verwendet Multi-Version Concurrency Control (MVCC), um gleichzeitige Transaktionen zu verwalten. Wenn Dokumente gelöscht oder aktualisiert werden, verbleiben ihre früheren Versionen als „tote“ Versionen in Sammlungen und Indizes. Der Garbage Collection-Prozess gibt automatisch Speicherplatz aus diesen toten Versionen für zukünftige Operationen zurück.

Eine Aufblähung des Indexes tritt auf, wenn die Indizes einer Sammlung aufgrund der Anhäufung von toten oder veralteten Indexeinträgen oder einer Fragmentierung innerhalb der Seiten größer werden. Der angegebene Prozentsatz steht für die Menge an Indexplatz, die von zukünftigen Indexeinträgen genutzt werden kann. Diese Überlastung verbraucht Speicherplatz sowohl im Puffercache als auch im Speicher. Wenn Sie das Aufblähen entfernen möchten, müssen Sie die Indizes neu erstellen.

Example Beispiel

Führen Sie den folgenden Befehl aus, um ungenutzten Speicherplatz für Ihren Index zu ermitteln:

```
db.coll.aggregate({$indexStats:{}});
```

Das Ergebnis sieht ungefähr so aus:

```
{
    "name" : "_id_",
```

```
    "key" : {
        "_id" : 1
    },
    "host" : "devbox-test.localhost.a2z.com:27317",
    "size" : NumberLong(827392),
    "accesses" : {
        "ops" : NumberLong(40000),
        "docsRead" : NumberLong(46049),
        "since" : ISODate("2025-04-03T21:44:51.251Z")
    },
    "cacheStats" : {
        "blksRead" : NumberLong(264),
        "blksHit" : NumberLong(140190),
        "hitRatio" : 99.8121
    },
    "unusedStorageSize" : {
        "unusedSizeBytes" : 409600,
        "unusedSizePercent" : 49.51
    }
}
```

Mit dem `reIndex` Befehl, der einen Scan der gesamten Sammlung erfordert, können Sie Indizes ohne Ausfallzeiten neu erstellen. Siehe Indexverwaltung mithilfe von `Reindex`.

Indexverwaltung mithilfe von **`Reindex`**

`reIndex`ist ein Befehl, der verwendet wird, um einen Index neu zu erstellen. Er wird normalerweise verwendet, wenn ein Index beschädigt oder ineffizient geworden ist. Im Laufe der Zeit können Indizes aufgrund vieler Aktualisierungen, Einfügungen oder Löschungen ungenutzten Speicherplatz ansammeln, was zu Leistungseinbußen führt. Die Neuindizierung hilft dabei, diesen ungenutzten Speicherplatz zu entfernen und die Effizienz des Indexes wiederherzustellen.

Richtlinien zur Neuindizierung

- `reIndex`wird nur auf Amazon DocumentDB 5.0 unterstützt.
- Amazon DocumentDB unterstützt `reindex` einen einzigen Index im Hintergrund, sodass mehrere Worker verwendet werden können. Der alte Index kann für Abfragen verwendet werden, wenn der `reIndex` Prozess läuft.
- Amazon DocumentDB unterstützt die Indexierung von Fortschrittsberichten über. `currentOp` Sie können die Phasen der Indexerstellung sehen, die denen bei der Phasen der Indexerstellung Indexerstellung ähneln. Der einzige Unterschied besteht darin, dass es `reIndex` immer acht

Stufen gibt, unabhängig davon, ob es sich um eine eindeutige Phase handelt oder nicht. Es gibt keine Phase „Index erstellen: Schlüssel 2 sortieren“.

- `reIndex`kann gleichzeitig mit jedem Befehl ausgeführt werden, mit Ausnahme von Befehlen, die sich auf den Index beziehen, für dieselbe Sammlung: `createIndexes``dropIndexes`, `collMod` und. `renameCollection`
- `reIndex`wird derzeit nicht für Text-, Geodaten-, Vektor- und Teilindizes unterstützt.

Erstellen Sie einen Neuindex

Verwenden Sie den folgenden Befehl, um Ihren Index neu zu erstellen:

```
db.runCommand({ reIndex: "collection-name", index: "index-name"})
```

Optional können Sie auch die Anzahl der Worker steuern, die dem Neuerstellungsprozess zugewiesen sind:

```
db.runCommand({ reIndex: "collection-name", index: "index-name", workers: number })
```

Verwaltung der Dokumentenkomprimierung auf Sammlungsebene

Mit der Dokumentenkomprimierung auf Sammlungsebene von Amazon DocumentDB 5.0 können Sie die Speicher- und I/O-Kosten senken, indem Sie die Dokumente in Ihren Sammlungen komprimieren. Sie können die Dokumentenkomprimierung auf Sammlungsebene aktivieren und bei Bedarf Komprimierungsmetriken anzeigen, indem Sie den Speicherzuwachs anhand von Komprimierungsmetriken wie der Speichergröße komprimierter Dokumente und dem Komprimierungsstatus messen. Amazon DocumentDB 5.0 verwendet den LZ4-Komprimierungsalgorithmus, um Dokumente zu komprimieren.

Amazon DocumentDB unterstützt die Dokumentenkomprimierung ab Version 5.0. Im Folgenden finden Sie Funktionen zur Komprimierung von Dokumenten auf Sammlungsebene:

- Standardverhalten — Die Standardkomprimierungseinstellung für neue Sammlungen auf einem 5.0-Cluster wird durch den Cluster-Parameter bestimmt. `default_collection_compression` Dieser Parameter ist standardmäßig auf „deaktiviert“ gesetzt.
- Komprimieren vorhandener Sammlungen — Die Komprimierungseinstellungen für bestehende Sammlungen können mit dem `collMod` Befehl geändert werden.

- Ändern des Komprimierungsschwellenwerts — Der Standardschwellenwert für die Komprimierung beträgt 2 KB. Dieser Wert kann für neue Sammlungen mit dem `createCollection` Befehl angegeben und für bestehende Sammlungen mit dem `collMod` Befehl geändert werden.

> **Note**
>
> Die Amazon DocumentDB DocumentDB-Dokumentenkomprimierung wird in den Amazon DocumentDB DocumentDB-Versionen 3.6 und 4.0 nicht unterstützt.

Themen

- Verwaltung der Dokumentenkomprimierung
- Überwachung der Dokumentenkomprimierung

Verwaltung der Dokumentenkomprimierung

Aktivieren der Dokumentenkomprimierung in einer Sammlung

Aktivieren Sie die Dokumentenkomprimierung beim Erstellen einer Sammlung in Amazon DocumentDB 5.0 mithilfe der folgenden `db.createCollection()` Methode:

```
db.createCollection( sample_collection,{
    storageEngine : {
        documentDB: {
            compression:{enable: <true | false>}
        }
    }
})
```

Aktivieren der Dokumentenkomprimierung in einem Cluster

Die Dokumentenkomprimierung kann standardmäßig für alle neuen Sammlungen auf Clusterebene aktiviert werden, indem der Cluster-Parameter `default_collection_compression` auf „enabled“ gesetzt wird. Wenn dieser Parameter auf „aktiviert“ gesetzt ist, ist für neu erstellte Sammlungen auf dem Cluster die Komprimierung standardmäßig aktiviert, wobei der Komprimierungsschwellenwert 2 KB beträgt.

Komprimieren vorhandener Sammlungen

Sie können auch die Komprimierungseinstellungen für eine vorhandene Sammlung ändern, indem Sie den `collMod` Vorgang verwenden und die folgende `storageEngine` Konfiguration angeben. Beachten Sie, dass die mit diesem Befehl vorgenommene Änderung nur für neue inserted/updated Dokumente gilt und dass sich die Komprimierung zuvor eingefügter Dokumente nicht ändert.

```
db.runCommand({
    collMod: "orders",
    storageEngine: {
        documentDB: {compression: {enable: <true | false>} }
    }
})
```

Einstellung der Schwellenwerte für die Komprimierung

Standardmäßig liegt der Komprimierungsschwellenwert für komprimierte Sammlungen bei 2032 Byte. Dieser Schwellenwert kann im `createCollection` Befehl festgelegt werden, wenn eine neue Sammlung mit aktivierter Komprimierung erstellt wird:

```
db.createCollection( sample_collection, {
    storageEngine : {
        documentDB: {
            compression: {
                enable: true,
                threshold: <128 - 8000>
            }
        }
    }
})
```

Sie können den Komprimierungsschwellenwert für eine vorhandene komprimierte Sammlung auch ändern, indem Sie den `collMod` Vorgang verwenden und die folgende `storageEngine` Konfiguration angeben:

```
db.runCommand({
    collMod: "orders",
    storageEngine: {
        documentDB: {
            compression: {
```

```
                enable: true,
                threshold: <128 - 8000>
            }
        }
    }
})
```

Beachten Sie, dass der Komprimierungsschwellenwert nur auf einen Wert zwischen 128 und 8000 Byte festgelegt werden kann. Außerdem muss die `enable` Option bei der Angabe des Komprimierungsschwellenwerts auf „true“ gesetzt werden.

Überwachung der Dokumentenkomprimierung

Sie können überprüfen, ob eine Sammlung komprimiert ist, und ihr Komprimierungsverhältnis wie folgt berechnen.

Zeigen Sie die Komprimierungsstatistiken an, indem Sie den `db.collection.stats()` Befehl `db.printCollectionStats()` oder in der Mongo-Shell ausführen. Die Ausgabe zeigt Ihnen die Originalgröße und die komprimierte Größe, die Sie vergleichen können, um die Speichergewinne durch die Dokumentenkomprimierung zu analysieren. In diesem Beispiel werden die Statistiken für eine Sammlung mit dem Namen „sample_collection“ unten angezeigt. Im Folgenden wird ein Skalierungsfaktor von 1024*1024 verwendet, um die Werte und in MB auszugeben. `size` `storageSize`

```
db.sample_collection.stats(1024*1024)
```

Das Folgende ist ein Beispiel für die Ausgabe für den obigen Befehl:

```
{
    "ns" : "test.sample_collection",
    "count" : 1000000,
    "size" : 3906.3,
    "avgObjSize" : 4096,
    "storageSize" : 1953.1,
    compression:{"enabled" : true,"threshold" : 2032},
    ...
}
```

- Größe — Die Originalgröße der Dokumentensammlung.

- avg ObjSize — Die durchschnittliche Dokumentgröße vor der Komprimierung, abgerundet auf die erste Dezimalzahl. Die Maßeinheit ist Byte.
- StorageSize — Die Speichergröße der Sammlung nach der Komprimierung. Die Maßeinheit ist Byte.
- aktiviert — Zeigt an, ob die Komprimierung aktiviert oder deaktiviert ist.

Um das tatsächliche Komprimierungsverhältnis zu berechnen, dividieren Sie die Sammlungsgröße durch die Speichergröße (size/storageSize). Für das obige Beispiel lautet die Berechnung 3906. 3/1953.1 was einem Kompressionsverhältnis von 2:1 entspricht.

Verwaltung der wörterbuchbasierten Komprimierung in Amazon DocumentDB 8.0

Amazon DocumentDB 8.0 führt einen neuen Algorithmus zur Dokumentenkomprimierung, zstd, als verbesserte Alternative zu LZ4 ein. Wenn Sie die Wörterbuchkomprimierung für eine Amazon DocumentDB 8.0-Sammlung aktivieren, indem Sie Zstd als Komprimierungsalgorithmus auswählen, werden Dokumente aus Ihrer Sammlung ausgewählt, um ein benutzerdefiniertes Komprimierungswörterbuch zu erstellen. Neue und aktualisierte Dokumente werden mit diesem Wörterbuch und dem zstd-Algorithmus komprimiert. Mit diesem Ansatz werden in der Regel höhere Komprimierungsraten erzielt als mit Standardkomprimierungsmethoden, insbesondere bei Sammlungen mit konsistenten Dokumentschemas oder wiederholten Feldnamen.

Lz4 ist ein Algorithmus, der für schnelle Komprimierung und Dekomprimierung entwickelt wurde. Es ist bekannt, dass er die CPU wenig belastet und gleichzeitig eine spürbare Komprimierung erreicht. Zstd ist ein Allzweckalgorithmus, der mit Standardeinstellungen mehr CPU verbraucht, aber bessere Komprimierungsraten als lz4 erzielt. Durch die Verwendung von Wörterbüchern wird die Komprimierung für die meisten JSON-Dokumente noch weiter verbessert. Einige Vorteile des Zstd-Algorithmus sind:

- Geringere Speicherkosten: Höhere Komprimierungsraten bedeuten weniger Speicherverbrauch und niedrigere Kosten.
- Niedriger I/O: Komprimierte Dokumente benötigen weniger I/O, was möglicherweise die Leistung verbessert.
- Optimiert für Ihre Sammlung: Das Wörterbuch wurde speziell für die Datenmuster Ihrer Sammlung trainiert.

> **Note**
>
> Dictionary-based Die Komprimierung wird in den Amazon DocumentDB DocumentDB-Versionen 3.6, 4.0 und 5.0 nicht unterstützt.

Leistungsaspekte

Die Zstd-Komprimierung beinhaltet die folgenden Kompromisse:

- Speicher im Vergleich zur CPU: Die Zstd-Komprimierung erzielt bessere Komprimierungsraten, verbraucht aber möglicherweise etwas mehr CPU-Ressourcen als die LZ4-Komprimierung.
- Anfängliche Komprimierung: Neue Sammlungen erreichen möglicherweise erst dann eine optimale Komprimierung, wenn genügend Dokumente eingefügt wurden, um ein effektives Wörterbuch zu trainieren. Derzeit wird ein Wörterbuch trainiert, wenn die Sammlung mindestens 100 Dokumente enthält.
- Workload-Typ: Bei Read-intensive Workloads, bei denen die gesamten Daten in den Puffercache passen, kann es aufgrund des Dekomprimierungsaufwands zu einem Anstieg der Latenz und der CPU-Auslastung kommen.

Die Zstd-Komprimierung ist besonders effektiv für Sammlungen mit kleinen Dokumenten, Dokumentarrays und sich wiederholenden Feldnamen.

Aktivierung der wörterbuchbasierten Komprimierung

Für neue Sammlungen können Sie den folgenden Befehl verwenden, um die Zstd-Komprimierung zu aktivieren:

```
db.createCollection("myCollection",
    {
        storageEngine: {
            documentDB: {
                compression: {
                    enable: true,
                    algorithm: "zstd"
                }
            }
```

```
        }
    }
)
```

Sie können die Komprimierung auch für eine bestehende Sammlung aktivieren oder ändern:

```
db.runCommand({
    collMod: "myCollection",
    storageEngine: {
        documentDB: {
            compression: {
                enable: true,
                algorithm: "zstd"
            }
        }
    }
})
```

Um den Zstd-Algorithmus für alle Sammlungen auf Ihrem Cluster zu aktivieren, können Sie die Cluster-Parametergruppe so ändern, dass „zstd“ als Wert für den Parameter „default_collection_compression“ ausgewählt wird.

Erste Schritte

Bei Amazon DocumentDB 8.0 ist die Zstd-Komprimierung standardmäßig aktiviert. Sie können sie jederzeit ausschalten, indem Sie den Wert von 'default_compression' in Ihrer Cluster-Parametergruppe auf disabled setzen. Es muss beachtet werden, dass ab Amazon DocumentDB 8.0 'enabled' keine gültige Wahl mehr für default_compression ist und Sie zwischen Zstd und LZ4 wählen müssen.

Überwachen

Sie können die Komprimierungsinformationen für eine Sammlung mit einem der folgenden Befehle anzeigen:

- db.runCommand ({collStats: „MyCollection“}) ODER
- db.collection.stats ()

Diese Befehle geben wichtige Statistiken zurück, anhand derer Sie das Kompressionsverhältnis berechnen können:

- compression.algorithm: Der verwendete Algorithmus („lz4" oder „zstd“)
- StorageSize: Der tatsächliche Speicherplatz, der von der Sammlung nach der Komprimierung verwendet wird. Beachten Sie, dass diese Zahl die Fragmentierung (d. h. den ungenutzten Speicherplatz auf Datenbankseiten) beinhaltet
- avgObjSize: Die durchschnittliche logische Größe der Dokumente der Sammlung, dekomprimiert. Beachten Sie: Wenn Ihre Sammlung mehr als 20.000 Dokumente umfasst, handelt es sich bei dieser Zahl um eine ungefähre Zahl, die auf einer Stichprobe von 20.000 Dokumenten basiert.
- Größe: Die logische Größe der Sammlung ohne Komprimierung. Diese Zahl ergibt sich aus der Multiplikation ObjSize von avg mit der Gesamtzahl der Dokumente in der Sammlung. Wenn avg also eine Näherung ObjSize ist, ist diese Zahl auch eine Näherung.
- Anzahl: Anzahl der Dokumente in der Sammlung

Die folgenden CloudWatch Messwerte können bei der Bewertung der wörterbuchbasierten Komprimierung hilfreich sein:

- CPUUtilization
- FreeableMemory
- VolumeBytesUsed
- VolumeReadIOPs
- VolumeWriteIOPs

CollStats-Metriken:

- storageSize
- size

Darüber hinaus kann es nützlich sein, anwendungsspezifische Kennzahlen wie Latenz und Durchsatz pro Abfragetyp oder API im Auge zu behalten.

Sicherheit in Amazon DocumentDB

Cloud-Sicherheit AWS hat höchste Priorität. Als AWS Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die auf die Anforderungen der sicherheitsrelevantesten Organisationen zugeschnitten sind.

Sicherheit ist eine gemeinsame Verantwortung von Ihnen AWS und Ihnen. Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung bei der Verwendung von Amazon DocumentDB anwenden. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist für den Schutz der Infrastruktur verantwortlich, auf der AWS Dienste in der AWS Cloud ausgeführt werden. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Third-party Auditoren testen und verifizieren regelmäßig die Wirksamkeit unserer Sicherheitsvorkehrungen im Rahmen der AWS Compliance-Programme. Weitere Informationen zu den Compliance-Programmen, die für Amazon DocumentDB gelten (mit MongoDB-Kompatibilität), finden Sie unter AWS Services in Scope by Compliance Program.
- Sicherheit in der Cloud — Ihre Verantwortung wird durch den AWS Service bestimmt, den Sie nutzen. In Ihre Verantwortung fallen außerdem weitere Faktoren, wie z. B. die Vertraulichkeit der Daten, die Anforderungen Ihrer Organisation sowie geltende Gesetze und Vorschriften.

Amazon DocumentDB ist im Rahmen des Federal Risk and Authorization Management Program (FedRAMP) autorisiert. Es verfügt über die FedRAMP High-Autorisierung für AWS GovCloud (US-) Regionen und die FedRAMP Moderate-Autorisierung für US-Regionen. AWS East/West Einzelheiten zu den Maßnahmen zur AWS Einhaltung der Vorschriften finden Sie unter AWS Services in Scope by Compliance Program.

> **Note**
>
> Dieses Kapitel gilt sowohl für instanzbasierte Cluster als auch für Elastic Clusters. Weitere Informationen finden Sie in den folgenden Themen.

Sie erfahren auch, wie Sie andere AWS Dienste nutzen können, mit denen Sie Ihre Amazon DocumentDB-Ressourcen überwachen und sichern können. Die folgenden Themen zeigen Ihnen, wie Sie Amazon DocumentDB so konfigurieren, dass Ihre Sicherheits- und Compliance-Ziele erreicht werden.

Topics

- Passwortverwaltung mit Amazon DocumentDB und AWS Secrets Manager
- Datenschutz in Amazon DocumentDB
- Identitäts- und Zugriffsmanagement für Amazon DocumentDB
- Cross-service verwirrt, stellvertretender Vertreter der Prävention
- Authentifizierung mithilfe der IAM-Identität
- Authentifizierung mit SCRAM-SHA-256
- Verwaltung von Amazon DocumentDB-Benutzern
- Datenbankzugriff mit Role-Based Access Control
- Aktualisierung Ihrer Amazon DocumentDB-TLS-Zertifikate
- Überprüfung der Konformität in Amazon DocumentDB
- Resilienz in Amazon DocumentDB
- Infrastruktursicherheit in Amazon DocumentDB
- Amazon DocumentDB-API und Schnittstellen-VPC-Endpunkte (AWS PrivateLink)
- Bewährte Sicherheitsmethoden für Amazon DocumentDB
- Prüfung von Amazon DocumentDB-Ereignissen
- Amazon VPC und Amazon DocumentDB

Passwortverwaltung mit Amazon DocumentDB und AWS Secrets Manager

Amazon DocumentDB ist in Secrets Manager integriert, um primäre Benutzerkennwörter für Ihre Cluster zu verwalten.

Topics

- Einschränkungen bei der Secrets Manager-Integration mit Amazon DocumentDB
- Überblick über die Verwaltung von Primärbenutzerkennwörtern mit AWS Secrets Manager
- Durchsetzung der Verwaltung des primären Benutzerpassworts durch Amazon DocumentDB in AWS Secrets Manager
- Verwaltung des primären Benutzerpassworts für einen Cluster mit Secrets Manager

Einschränkungen bei der Secrets Manager-Integration mit Amazon DocumentDB

Die Verwaltung von Primärbenutzerkennwörtern mit Secrets Manager wird für die folgenden Funktionen nicht unterstützt:

- Cluster, die Teil einer globalen Amazon DocumentDB-Datenbank sind
- Amazon DocumentDB Regionsübergreifende Read Replicas

Überblick über die Verwaltung von Primärbenutzerkennwörtern mit AWS Secrets Manager

Mit AWS Secrets Manager können Sie hartcodierte Anmeldeinformationen in Ihrem Code, einschließlich Datenbankkennwörter, durch einen API-Aufruf an Secrets Manager ersetzen, um das Geheimnis programmgesteuert abzurufen. Weitere Informationen zu Secrets Manager finden Sie im Benutzerhandbuch für AWS Secrets Manager.

Wenn Sie Datenbankgeheimnisse in Secrets Manager speichern, fallen für Ihr AWS Konto Gebühren an. Informationen zu Preisen finden Sie unter AWS Secrets Manager -Preise.

Sie können angeben, dass Amazon DocumentDB das primäre Benutzerkennwort in Secrets Manager für einen Amazon DocumentDB-Cluster verwaltet, wenn Sie eine der folgenden Operationen ausführen:

- Den Cluster erstellen
- Ändern Sie den Cluster

Wenn Sie angeben, dass Amazon DocumentDB das primäre Benutzerkennwort in Secrets Manager verwaltet, generiert Amazon DocumentDB das Passwort und speichert es in Secrets Manager. Sie können direkt mit dem Secret interagieren, um die Anmeldeinformationen für den Hauptbenutzer abzurufen. Sie können auch einen vom Kunden verwalteten Schlüssel angeben, um das Geheimnis zu verschlüsseln, oder den KMS-Schlüssel verwenden, den Secrets Manager bereitstellt.

Amazon DocumentDB verwaltet die Einstellungen für das Geheimnis und wechselt das Geheimnis standardmäßig alle sieben Tage. Sie können einige Einstellungen ändern, wie zum Beispiel den Rotationsplan. Wenn Sie einen Cluster löschen, der ein Geheimnis in Secrets Manager verwaltet, werden das Geheimnis und die zugehörigen Metadaten ebenfalls gelöscht.

Um mit den Anmeldeinformationen in einem Secret eine Verbindung zu einem Cluster herzustellen, können Sie das Secret aus Secrets Manager abrufen. Weitere Informationen finden Sie unter Abrufen von Geheimnissen aus einer SQL-Datenbank AWS Secrets Manager und Herstellen einer Verbindung zu einer SQL-Datenbank mithilfe von JDBC mit Anmeldeinformationen in einem AWS Secrets Manager geheimen Schlüssel im AWS Secrets Manager Benutzerhandbuch.

Durchsetzung der Verwaltung des primären Benutzerpassworts durch Amazon DocumentDB in AWS Secrets Manager

Sie können IAM-Bedingungsschlüssel verwenden, um die Verwaltung des Primärbenutzerkennworts durch Amazon DocumentDB in durchzusetzen. AWS Secrets Manager Die folgende Richtlinie erlaubt Benutzern nicht, Instances oder Cluster zu erstellen oder wiederherzustellen, es sei denn, das primäre Benutzerkennwort wird von Amazon DocumentDB in Secrets Manager verwaltet.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "rds:CreateDBCluster"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "rds:ManageMasterUserPassword": false
                }
            }
        }
    ]
}
```

Verwaltung des primären Benutzerpassworts für einen Cluster mit Secrets Manager

Sie können die Amazon DocumentDB-Verwaltung des Primärbenutzerkennworts in Secrets Manager konfigurieren, wenn Sie die folgenden Aktionen ausführen:

- Erstellen eines Amazon DocumentDB-Clusters
- Ändern eines Amazon DocumentDB-Clusters

Sie können die Amazon DocumentDB-Konsole oder die verwenden AWS CLI , um diese Aktionen durchzuführen.

Datenschutz in Amazon DocumentDB

Das Modell der AWS gemeinsamen Verantwortung (Modell der gilt für den Datenschutz in Amazon DocumentDB. Wie in diesem Modell beschrieben, AWS ist es für den Schutz der globalen Infrastruktur verantwortlich, auf der AWS Cloud alle ausgeführt werden. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie in den häufig gestellten Fragen zum Datenschutz in der Region . Weitere Informationen zum Datenschutz in Europa finden Sie im Zentrum für die Datenschutz-Grundverordnung (DSGVO).

Aus Datenschutzgründen empfehlen wir, Ihre AWS-Konto Anmeldeinformationen zu schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einzurichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Wird verwendet SSL/TLS , um mit AWS Ressourcen zu kommunizieren. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein AWS CloudTrail. Informationen zur Verwendung von CloudTrail Pfaden zur Erfassung von AWS Aktivitäten findest du unter Arbeiten mit CloudTrail Pfaden im AWS CloudTrail Benutzerhandbuch.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.

- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-3-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-3.

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit Amazon DocumentDB oder anderen AWS-Services über die Konsole, die API oder SDKs arbeiten. AWS CLI AWS Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Themen

- Client-side Verschlüsselung auf Feldebene
- Verschlüsselung von Amazon DocumentDB-Daten im Ruhezustand
- Verschlüsseln von Daten während der Übertragung
- Schlüsselverwaltung

Client-side Verschlüsselung auf Feldebene

Mit der clientseitigen Verschlüsselung auf Feldebene (FLE) von Amazon DocumentDB können Sie vertrauliche Daten in Ihren Client-Anwendungen verschlüsseln, bevor sie an einen Amazon DocumentDB-Cluster übertragen werden. Sensible Daten bleiben verschlüsselt, wenn sie in einem Cluster gespeichert und verarbeitet werden, und werden beim Abrufen in der Client-Anwendung entschlüsselt.

Topics

- Erste Schritte
- Abfragen in einer clientseitigen FLE

Erste Schritte

Die Erstkonfiguration der clientseitigen FLE in Amazon DocumentDB ist ein vierstufiger Prozess, der die Erstellung eines Verschlüsselungsschlüssels, das Zuweisen einer Rolle zur Anwendung, die Konfiguration der Anwendung und die Definition des CRUD-Vorgangs mit Verschlüsselungsoptionen umfasst.

Topics

- Schritt 1: Erstellen Sie die Verschlüsselungsschlüssel
- Schritt 2: Ordnen Sie der Anwendung eine Rolle zu
- Schritt 3: Konfigurieren Sie die Anwendung
- Schritt 4: Definieren Sie eine CRUD-Operation
- Beispiel: Konfigurationsdatei für die clientseitige Verschlüsselung auf Feldebene

Schritt 1: Erstellen Sie die Verschlüsselungsschlüssel

Verwenden Sie: Erstellen Sie einen symmetrischen AWS Key Management Service, vom Kunden verwalteten Schlüssel, der das sensible Datenfeld verschlüsselt und entschlüsselt, und gewähren Sie ihm die erforderlichen IAM-Nutzungsberechtigungen. AWS KMS speichert den vom Kunden verwalteten Schlüssel, der zum Verschlüsseln von Datenschlüsseln verwendet wird. Durch das Speichern des vom Kunden verwalteten Schlüssels wird Ihre Sicherheitslage AWS KMS gestärkt. Der Datenschlüssel ist der Sekundärschlüssel, der in einer Amazon DocumentDB-Sammlung gespeichert wird und benötigt wird, um vertrauliche Felder zu verschlüsseln, bevor das Dokument in Amazon DocumentDB gespeichert wird. Der vom Kunden verwaltete Schlüssel verschlüsselt den Datenschlüssel, der wiederum Ihre Daten verschlüsselt und entschlüsselt. Wenn Sie einen globalen Cluster verwenden, können Sie einen Schlüssel für mehrere Regionen erstellen, den verschiedene Servicerollen in verschiedenen Regionen verwenden können.

Weitere Informationen dazu AWS Key Management Service, einschließlich der Erstellung eines Schlüssels, finden Sie unter AWS Key Management Service Konzepte im AWS Key Management Service Entwicklerhandbuch.

Schritt 2: Ordnen Sie der Anwendung eine Rolle zu

Erstellen Sie eine IAM-Richtlinie mit den entsprechenden AWS KMS Berechtigungen. Diese Richtlinie ermöglicht es den IAM-Identitäten, an die sie angehängt ist, den im Ressourcenfeld angegebenen KMS-Schlüssel zu verschlüsseln und zu entschlüsseln. Ihre Anwendung übernimmt diese IAM-Rolle, mit der sie sich authentifizieren kann. AWS KMS

Die Richtlinie sollte in etwa wie folgt aussehen:

```
{ "Effect": "Allow",
"Action": ["kms:Decrypt", "kms:Encrypt"],
"Resource": "Customer Key ARN"
}
```

Schritt 3: Konfigurieren Sie die Anwendung

Sie haben einen vom Kunden verwalteten Schlüssel definiert AWS KMS und eine IAM-Rolle mit den erforderlichen Zugriffsberechtigungen erstellt. Importieren Sie die erforderlichen Pakete.

```
import boto3
import json
import base64
from pymongo import MongoClient
from pymongo.encryption import (Algorithm,
                                ClientEncryption)
```

```
# create a session object:
my_session = boto3.session.Session()

# get access_key and secret_key programmatically using get_frozen_credentials() method:
current_credentials = my_session.get_credentials().get_frozen_credentials()
```

1. Geben Sie aws als AWS KMS Anbietertyp an und geben Sie die Kontoanmeldeinformationen ein, die Sie im vorherigen Schritt abgerufen haben.

   ```
   provider = "aws"
   kms_providers = {
       provider: {
           "accessKeyId": current_credentials.access_key,
           "secretAccessKey": current_credentials.secret_key
       }
   }
   ```

2. Geben Sie den vom Kunden verwalteten Schlüssel zum Verschlüsseln des Datenschlüssels an:

   ```
   customer_key = {
   “region”: “AWS region of the customer_key”,
       “key”: “customer_key ARN”
   ```

```
}

key_vault_namespace = "encryption.dataKeys"

key_alt_name = 'TEST_DATA_KEY'
```

3. Konfigurieren Sie das MongoClient Objekt:

```
client = MongoClient(connection_string)

coll = client.test.coll
coll.drop()

client_encryption = ClientEncryption(
    kms_providers, # pass in the kms_providers variable from the previous step
    key_vault_namespace = key_vault_namespace,
    client,
    coll.codec_options
)
```

4. Generieren Sie Ihren Datenschlüssel:

```
data_key_id = client_encryption.create_data_key(provider,
    customer_key,
    key_alt_name = [key_alt_name])
```

5. Rufen Sie Ihren vorhandenen Datenschlüssel ab:

```
data_key = DataKey("aws",
    master_key = customer_key)
key_id = data_key["_id"]
data_key_id = client[key_vault_namespace].find_one({"_id": key_id})
```

Schritt 4: Definieren Sie eine CRUD-Operation

Definieren Sie den CRUD-Vorgang mit Verschlüsselungsoptionen.

1. Definieren Sie die Sammlung, um ein einzelnes Dokument zu write/read /löschen:

```
coll = client.gameinfo.users
```

2. Explizite Verschlüsselung — Felder verschlüsseln und Folgendes einfügen:

> **Note**
>
> Es muss genau einer der Werte „key_id“ oder „key_alt_name“ angegeben werden.

```
encrypted_first_name = client_encryption.encrypt(
    "Jane",
    Algorithm.AEAD_AES_256_CBC_HMAC_SHA_512_Deterministic,
    key_id=data_key_id
)
encrypted_last_name = client_encryption.encrypt(
    "Doe",
    Algorithm.AEAD_AES_256_CBC_HMAC_SHA_512_Deterministic,
    key_id=data_key_id
)
encrypted_dob = client_encryption.encrypt(
    "1990-01-01",
    Algorithm.AEAD_AES_256_CBC_HMAC_SHA_512_Random,
    key_id=data_key_id
)

coll.insert_one(
    {"gamerTag": "jane_doe90",
    "firstName": encrypted_first_name,
    "lastName": encrypted_last_name,
    "dateOfBirth":encrypted_dob,
    "Favorite_games":["Halo","Age of Empires 2","Medal of Honor"]
})
```

Beispiel: Konfigurationsdatei für die clientseitige Verschlüsselung auf Feldebene

Ersetzen Sie im folgenden Beispiel jede Datei *placeholder* durch Ihre eigenen Informationen.

```
# import python packages:
import boto3
import json
import base64
from pymongo import MongoClient
from pymongo.encryption import (Algorithm,
                                ClientEncryption)
```

```
def main():

    # create a session object:
    my_session = boto3.session.Session()

    # get aws_region from session object:
    aws_region = my_session.region_name

    # get access_key and secret_key programmatically using get_frozen_credentials()
 method:
    current_credentials = my_session.get_credentials().get_frozen_credentials()
    provider = "aws"

    # define the kms_providers which is later used to create the Data Key:
    kms_providers = {
        provider: {
            "accessKeyId": current_credentials.access_key,
            "secretAccessKey": current_credentials.secret_key
        }
    }

    # enter the kms key ARN. Replace the example ARN value.
    kms_arn = "arn:aws:kms:us-east-1:123456789:key/abcd-efgh-ijkl-mnop"
    customer_key = {
        "region": aws_region,
        "key":kms_arn
    }

    # secrets manager is used to store and retrieve user credentials for connecting to
 an Amazon DocumentDB cluster.
    # retrieve the secret using the secret name. Replace the example secret key.
    secret_name = "/dev/secretKey"
    docdb_credentials = json.loads(my_session.client(service_name = 'secretsmanager',
 region_name = "us-east-1").get_secret_value(SecretId = secret_name)['SecretString'])

    connection_params = '/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false'
    conn_str = 'mongodb://' + docdb_credentials["username"] + ':' +
 docdb_credentials["password"] + '@' + docdb_credentials["host"] + ':' +
 str(docdb_credentials["port"]) + connection_params
    client = MongoClient(conn_str)

    coll = client.test.coll
```

```
    coll.drop()

    # store the encryption data keys in a key vault collection (having naming
 convention as db.collection):
    key_vault_namespace = "encryption.dataKeys"
    key_vault_db_name, key_vault_coll_name = key_vault_namespace.split(".", 1)

    # set up the key vault (key_vault_namespace) for this example:
    key_vault = client[key_vault_db_name][key_vault_coll_name]
    key_vault.drop()
    key_vault.create_index("keyAltNames", unique=True)

    client_encryption = ClientEncryption(
        kms_providers,
        key_vault_namespace,
        client,
        coll.codec_options)

    # create a new data key for the encrypted field:
    data_key_id = client_encryption.create_data_key(provider, master_key=customer_key,
 key_alt_names=["some_key_alt_name"], key_material = None)

    # explicitly encrypt a field:
    encrypted_first_name = client_encryption.encrypt(
    "Jane",
    Algorithm.AEAD_AES_256_CBC_HMAC_SHA_512_Deterministic,
    key_id=data_key_id
    )
    coll.insert_one(
    {"gamerTag": "jane_doe90",
    "firstName": encrypted_first_name
    })
    doc = coll.find_one()
    print('Encrypted document: %s' % (doc,))

    # explicitly decrypt the field:
    doc["firstName"] = client_encryption.decrypt(doc["firstName"])
    print('Decrypted document: %s' % (doc,))

    # cleanup resources:
    client_encryption.close()
    client.close()

if __name__ == "__main__":
```

```
    main()
```

Abfragen in einer clientseitigen FLE

Amazon DocumentDB unterstützt nur Gleichheitsabfragen mit clientseitigen FLE-verschlüsselten Werten.

Um beispielsweise Dokumente abzufragen, bei denen der verschlüsselte Gamerscore 500 entspricht, verwendet der Client eine explizite Verschlüsselungsmethode, um den Abfragewert zu verschlüsseln:

```
encrypted_gamerscore_filter = client_encryption.encrypt(
    500,
    Algorithm.AEAD_AES_256_CBC_HMAC_SHA_512_Deterministic,
    key_id=data_key_id
)

coll.find( {
    "gamerscore" : { "$eq" : encrypted_gamerscore_filter }
} )
```

Verschlüsselung von Amazon DocumentDB-Daten im Ruhezustand

> **Note**
>
> AWS KMS ersetzt den Begriff Customer Master Key (CMK) durch AWS KMS key einen KMS-Schlüssel. Das Konzept hat sich nicht geändert. Um grundlegende Änderungen zu vermeiden, AWS KMS werden einige Varianten dieses Begriffs beibehalten.

Sie verschlüsseln ruhende Daten in Ihrem Amazon DocumentDB-Cluster, indem Sie bei der Erstellung Ihres Clusters die Speicherverschlüsselungsoption angeben. Die Speicher-Verschlüsselung ist Cluster-weit aktiviert und wird auf alle Instances angewendet, einschließlich der primären Instance und alle Replikate. Sie wird auch auf das Speichervolumen Ihres Clusters, auf Daten, Indizes, Protokolle, automatisierte Sicherungen und Snapshots angewendet.

Amazon DocumentDB verwendet den 256-Bit-Advanced Encryption Standard (AES-256), um Ihre Daten mit den in () gespeicherten Verschlüsselungsschlüsseln zu verschlüsseln. AWS Key Management Service AWS KMS Wenn Sie einen Amazon DocumentDB-Cluster mit aktivierter Verschlüsselung im Ruhezustand verwenden, müssen Sie Ihre Anwendungslogik oder Client-

Verbindung nicht ändern. Amazon DocumentDB wickelt die Verschlüsselung und Entschlüsselung Ihrer Daten transparent ab, mit minimalen Auswirkungen auf die Leistung.

Amazon DocumentDB integriert AWS KMS und verwendet eine Methode, die als Umschlagverschlüsselung bekannt ist, um Ihre Daten zu schützen. Wenn ein Amazon DocumentDB-Cluster mit einem verschlüsselt ist AWS KMS, bittet Amazon DocumentDB darum, Ihren KMS-Schlüssel AWS KMS zu verwenden, um einen Chiffretext-Datenschlüssel zur Verschlüsselung des Speichervolumes zu generieren. Der Chiffretext-Datenschlüssel wird mit dem von Ihnen definierten KMS-Schlüssel verschlüsselt und zusammen mit den verschlüsselten Daten und Speichermetadaten gespeichert. Wenn Amazon DocumentDB auf Ihre verschlüsselten Daten zugreifen muss, fordert AWS KMS es die Entschlüsselung des Chiffretext-Datenschlüssels mithilfe Ihres KMS-Schlüssels an und speichert den Klartext-Datenschlüssel im Speicher, um Daten auf dem Speichervolume effizient zu verschlüsseln und zu entschlüsseln.

Die Speicherverschlüsselungsfunktion in Amazon DocumentDB ist für alle unterstützten Instance-Größen verfügbar und überall dort, wo Amazon DocumentDB verfügbar ist. AWS-Regionen

Aktivierung der Verschlüsselung im Ruhezustand für einen Amazon DocumentDB-Cluster

Sie können die Verschlüsselung im Ruhezustand auf einem Amazon DocumentDB-Cluster aktivieren oder deaktivieren, wenn der Cluster entweder mit AWS-Managementkonsole oder mit AWS Command Line Interface ()AWS CLI bereitgestellt wird. Cluster, die Sie mit der Konsole erstellen, haben standardmäßig die Verschlüsselung im Ruhezustand aktiviert. Bei Clustern, die Sie mit dem erstellen, ist AWS CLI die Verschlüsselung im Ruhezustand standardmäßig deaktiviert. Daher müssen Sie die Verschlüsselung im Ruhezustand explizit mit dem `--storage-encrypted`-Parameter aktivieren. In beiden Fällen können Sie nach dem Erstellen des Clusters die Option „Verschlüsselung im Ruhezustand“ nicht ändern.

Amazon DocumentDB verwendet es, AWS KMS um Verschlüsselungsschlüssel abzurufen und zu verwalten und um die Richtlinien zu definieren, die steuern, wie diese Schlüssel verwendet werden können. Wenn Sie keine AWS KMS Schlüssel-ID angeben, verwendet Amazon DocumentDB den standardmäßigen KMS-Schlüssel für AWS verwaltete Dienste. Amazon DocumentDB erstellt für jeden AWS-Region in Ihrem einen eigenen KMS-Schlüssel. AWS-Konto Weitere Informationen finden Sie unter AWS Key Management Service -Konzepte.

Informationen zum Erstellen Ihres eigenen KMS-Schlüssels finden Sie unter Erste Schritte im AWS Key Management Service Entwicklerhandbuch.

> **Important**
>
> Sie müssen einen KMS-Schlüssel für die symmetrische Verschlüsselung verwenden, um Ihren Cluster zu verschlüsseln, da Amazon DocumentDB nur KMS-Schlüssel mit symmetrischer Verschlüsselung unterstützt. Verwenden Sie keinen asymmetrischen KMS-Schlüssel, um zu versuchen, die Daten in Ihren Amazon DocumentDB-Clustern zu verschlüsseln. Weitere Informationen finden Sie im AWS KMS Entwicklerhandbuch unter Asymmetrische Schlüssel. AWS Key Management Service

Wenn Amazon DocumentDB nicht mehr auf den KMS-Schlüssel für einen Cluster zugreifen kann — wenn beispielsweise der Eigentümer des AWS-Konto Schlüssels gesperrt ist, der Schlüssel deaktiviert ist, das Löschen des Schlüssels geplant ist oder die Schlüsselrichtlinie oder -gewährung, auf die Amazon DocumentDB angewiesen ist, entfernt wird — geht der Cluster zuerst in den Status über. `inaccessible-encryption-credentials-recoverable` Solange sich der Cluster in diesem Status befindet, stoppt Amazon DocumentDB die Instances des Clusters und Sie können nicht aus dem Cluster lesen oder in den Cluster schreiben. Der Cluster kann jedoch trotzdem wiederhergestellt werden, wenn der Zugriff auf den KMS-Schlüssel innerhalb von 7 Tagen wiederhergestellt wird. Wenn der Zugriff nicht innerhalb von 7 Tagen wiederhergestellt wird, wechselt der Cluster in den `inaccessible-encryption-credentials` Terminal-Status. Ab dem Terminalstatus ist der Cluster nicht mehr verfügbar und der aktuelle Status der Datenbank kann nicht wiederhergestellt werden. Sie können die Wiederherstellung nur aus einer Sicherung durchführen oder eine Point-in-Time-Wiederherstellung mit dem ursprünglichen KMS-Schlüssel durchführen. Für Amazon DocumentDB sind Backups immer für mindestens einen Tag aktiviert.

> **Note**
>
> Cluster, die Teil eines globalen Clusters sind, verhalten sich anders. Wenn Amazon DocumentDB feststellt, dass es nicht mehr auf den KMS-Schlüssel zugreifen kann, gehen alle Cluster im globalen Cluster direkt in den `inaccessible-encryption-credentials` Terminalstatus über und überspringen den Status „Wiederherstellbar“. Dies liegt daran, dass ein Cluster, der Teil eines globalen Clusters ist, nur gestoppt und gestartet werden kann, wenn er der einzige Cluster im globalen Cluster ist. Zur Wiederherstellung müssen Sie die Daten anhand eines Snapshots wiederherstellen oder eine Point-in-Time-Wiederherstellung durchführen. Um die ursprünglichen Cluster zu löschen, müssen Sie zuerst jeden Cluster aus dem globalen Cluster entfernen und ihn dann löschen.

> **Important**
>
> Sie können den KMS-Schlüssel für einen verschlüsselten Cluster nicht ändern, nachdem Sie ihn bereits erstellt haben. Stellen Sie sicher, dass Sie Ihre Anforderungen an den Verschlüsselungsschlüssel bestimmen, bevor Sie Ihren verschlüsselten Cluster erstellen.

Using the AWS-Managementkonsole

Geben Sie beim Erstellen eines Clusters die Option „Verschlüsselung im Ruhezustand“ an. Die Verschlüsselung im Ruhezustand ist standardmäßig aktiviert, wenn Sie einen Cluster mit der AWS-Managementkonsole erstellen. Nachdem der Cluster erstellt wurde, kann er nicht mehr geändert werden.

So legen Sie beim Erstellen eines Clusters die Option „Verschlüsselung im Ruhezustand“ fest

1. Erstellen Sie einen Amazon DocumentDB-Cluster, wie im Abschnitt Erste Schritte beschrieben. Wählen Sie jedoch in Schritt 6 nicht Create Cluster (Cluster erstellen) aus.
2. Wählen Sie unterhalb des Abschnitts Authentication (Authentifizierung) die Option Show advanced settings (Erweiterte Einstellungen anzeigen) aus.
3. Scrollen Sie nach unten zum Encryption-at-rest Abschnitt.
4. Wählen Sie die Option aus, die Sie für die Verschlüsselung im Ruhezustand wünschen. Egal, welche Option Sie wählen, nach dem Erstellen des Clusters können Sie sie nicht mehr ändern.
   - Wenn Sie Daten im Ruhezustand in diesem Cluster verschlüsseln möchten, wählen Sie Enable encryption (Verschlüsselung aktivieren) aus.
   - Wenn Sie Daten im Ruhezustand in diesem Cluster nicht verschlüsseln möchten, wählen Sie Disable encryption (Verschlüsselung deaktivieren) aus.
5. Wählen Sie den gewünschten Primärschlüssel aus. Amazon DocumentDB verwendet AWS Key Management Service (AWS KMS), um Verschlüsselungsschlüssel abzurufen und zu verwalten und um die Richtlinien zu definieren, die steuern, wie diese Schlüssel verwendet werden können. Wenn Sie keine AWS KMS Schlüssel-ID angeben, verwendet Amazon DocumentDB den standardmäßigen KMS-Schlüssel für AWS verwaltete Dienste. Weitere Informationen finden Sie unter AWS Key Management Service -Konzepte.

> **Note**
>
> Nachdem Sie einen verschlüsselten Cluster erstellt haben, können Sie den KMS-Schlüssel für diesen Cluster nicht mehr ändern. Stellen Sie sicher, dass Sie Ihre Anforderungen an den Verschlüsselungsschlüssel bestimmen, bevor Sie Ihren verschlüsselten Cluster erstellen.

6. Füllen Sie die anderen Abschnitte nach Bedarf aus und erstellen Sie Ihren Cluster.

Using the AWS CLI

Um einen Amazon DocumentDB-Cluster mit dem zu verschlüsseln AWS CLI, führen Sie den `create-db-cluster` Befehl aus und geben Sie die `--storage-encrypted` Option an. Amazon DocumentDB-Cluster, die mit der Option erstellt wurden, aktivieren standardmäßig AWS CLI keine Speicherverschlüsselung.

Im folgenden Beispiel wird ein Amazon DocumentDB-Cluster mit aktivierter Speicherverschlüsselung erstellt.

Ersetzen Sie in den folgenden Beispielen jedes *`user input placeholder`* durch die Informationen Ihres Clusters.

Example

Für Linux, macOS oder Unix:

```
aws docdb create-db-cluster \
  --db-cluster-identifier mydocdbcluster \
  --port 27017 \
  --engine docdb \
  --master-username SampleUser1 \
  --master-user-password primaryPassword \
  --storage-encrypted
```

Für Windows:

```
aws docdb create-db-cluster ^
  --db-cluster-identifier SampleUser1 ^
```

```
  --port 27017 ^
  --engine docdb ^
  --master-username SampleUser1 ^
  --master-user-password primaryPassword ^
  --storage-encrypted
```

Wenn Sie einen verschlüsselten Amazon DocumentDB-Cluster erstellen, können Sie wie im folgenden Beispiel einen AWS KMS Schlüsselbezeichner angeben.

Example

Für Linux, macOS oder Unix:

```
aws docdb create-db-cluster \
  --db-cluster-identifier SampleUser1 \
  --port 27017 \
  --engine docdb \
  --master-username primaryUsername \
  --master-user-password yourPrimaryPassword \
  --storage-encrypted \
  --kms-key-id key-arn-or-alias
```

Für Windows:

```
aws docdb create-db-cluster ^
  --db-cluster-identifier SampleUser1 ^
  --port 27017 ^
  --engine docdb ^
  --master-username SampleUser1 ^
  --master-user-password primaryPassword ^
  --storage-encrypted ^
  --kms-key-id key-arn-or-alias
```

> **Note**
>
> Nachdem Sie einen verschlüsselten Cluster erstellt haben, können Sie den KMS-Schlüssel für diesen Cluster nicht mehr ändern. Stellen Sie sicher, dass Sie Ihre Anforderungen an den Verschlüsselungsschlüssel bestimmen, bevor Sie Ihren verschlüsselten Cluster erstellen.

Auflösen eines Amazon DocumentDB-Clusters in einem Verschlüsselungszustand, auf den nicht zugegriffen werden kann

Ein Amazon DocumentDB-Cluster wechselt in den Verschlüsselungsstatus „Unzugänglich“, wenn Amazon DocumentDB nicht auf den KMS-Schlüssel zugreifen kann, mit dem der Cluster verschlüsselt wurde. Es gibt zwei solcher Status, und der Wiederherstellungspfad hängt davon ab, in welchem Status sich der Cluster befindet.

`Status „Nicht zugreifbar – Verschlüsselung – Anmeldeinformationen – wiederherstellbar“`

Solange sich der Cluster im Status befindet, können Sie den Cluster auf den `inaccessible-encryption-credentials-recoverable` Status zurücksetzen, `available` indem Sie den Zugriff von Amazon DocumentDB auf den KMS-Schlüssel wiederherstellen und dann den Cluster starten. Gehen Sie wie folgt vor, um diesen Status zu beheben:

1. Vergewissern Sie sich AWS-Konto , dass der, dem der KMS-Schlüssel gehört, aktiv ist. Wenn das Konto gesperrt ist, reaktivieren Sie es.
2. Vergewissern Sie sich, dass der KMS-Schlüssel aktiviert ist. Weitere Informationen finden Sie im AWS Key Management Service Entwicklerhandbuch unter Schlüssel aktivieren und deaktivieren.
3. Prüfen Sie, ob das Löschen des KMS-Schlüssels geplant ist. Ist dies der Fall, brechen Sie das geplante Löschen des Schlüssels ab. Weitere Informationen finden Sie im AWS Key Management Service Entwicklerhandbuch unter Planen und Abbrechen des Löschens von Schlüsseln.
4. Vergewissern Sie sich, dass Amazon DocumentDB aufgrund der KMS-Schlüsselrichtlinie und aller Zuschüsse, auf die Amazon DocumentDB angewiesen ist, den Schlüssel weiterhin verwenden darf.
5. Nachdem der Zugriff auf den KMS-Schlüssel wiederhergestellt ist, starten Sie den Cluster mithilfe des Befehls AWS-Managementkonsole oder, indem Sie den `start-db-cluster` AWS CLI Befehl ausführen.

   Example

   Für Linux, macOS oder Unix:

   ```
   aws docdb start-db-cluster \
     --db-cluster-identifier example-cluster
   ```

   Für Windows:

```
aws docdb start-db-cluster ^
  --db-cluster-identifier example-cluster
```

> **Important**
>
> Wenn der Zugriff auf den KMS-Schlüssel nicht innerhalb von 7 Tagen wiederhergestellt wird, wechselt der Cluster in den `inaccessible-encryption-credentials` Terminalstatus, von dem aus er nicht gestartet werden kann.

`Status: Anmeldeinformationen für Verschlüsselung nicht zugänglich`

Der `inaccessible-encryption-credentials` Status ist Terminal. Der Cluster kann nicht gestartet werden und der Betriebsstatus der Datenbank kann nicht wiederhergestellt werden. Um Ihre Daten wiederherzustellen, stellen Sie sie aus einem Snapshot wieder her oder führen Sie eine Point-in-Time-Wiederherstellung in einem neuen Cluster durch. Sie müssen weiterhin Zugriff auf den ursprünglichen KMS-Schlüssel haben, um die Wiederherstellung durchzuführen. Wenn der KMS-Schlüssel gelöscht wurde, können die Daten nicht wiederhergestellt werden.

Weitere Informationen erhalten Sie unter Wiederherstellung aus einem Cluster-Snapshot und Wiederherstellung auf einen bestimmten Zeitpunkt.

> **Note**
>
> Cluster, die Teil eines globalen Clusters sind, gehen direkt in den `inaccessible-encryption-credentials` Status über, wenn der Zugriff auf den KMS-Schlüssel verloren geht, da ein Cluster, der Teil eines globalen Clusters ist, nur gestoppt und gestartet werden kann, wenn er der einzige Cluster im globalen Cluster ist. Um einen Cluster zu löschen, der sich in diesem Status befindet, entfernen Sie zuerst jeden Cluster aus dem globalen Cluster und löschen Sie dann die Cluster einzeln.

Wenn Sie einen Cluster, der sich im `inaccessible-encryption-credentials` Status befindet, nicht löschen können, weil der Löschschutz aktiviert ist, deaktivieren Sie den Löschschutz, indem Sie den verwenden, AWS CLI bevor Sie den Löschvorgang erneut versuchen.

Example

Für Linux, macOS oder Unix:

```
aws docdb modify-db-cluster \
  --db-cluster-identifier example-cluster \
  --no-deletion-protection
```

Für Windows:

```
aws docdb modify-db-cluster ^
  --db-cluster-identifier example-cluster ^
  --no-deletion-protection
```

Anschließend können Sie den Cluster mithilfe des `delete-db-cluster` Befehls löschen.

Example

Für Linux, macOS oder Unix:

```
aws docdb delete-db-cluster \
  --db-cluster-identifier example-cluster \
  --skip-final-snapshot
```

Für Windows:

```
aws docdb delete-db-cluster ^
  --db-cluster-identifier example-cluster ^
  --skip-final-snapshot
```

Wenn der Cluster nicht gelöscht wird, nachdem Sie die vorherigen Befehle ausgeführt haben, wenden Sie sich an AWS Support.

Einschränkungen für mit Amazon DocumentDB verschlüsselte Cluster

Für mit Amazon DocumentDB verschlüsselte Cluster gelten die folgenden Einschränkungen.

- Sie können die Verschlüsselung im Ruhezustand für einen Amazon DocumentDB-Cluster nur zum Zeitpunkt der Erstellung aktivieren oder deaktivieren, nicht nachdem der Cluster erstellt wurde. Sie können jedoch eine verschlüsselte Kopie eines unverschlüsselten Clusters erstellen, indem

Sie einen Snapshot des unverschlüsselten Clusters erstellen und dann den unverschlüsselten Snapshot als neuen Cluster wiederherstellen und dabei die Option Verschlüsselung im Ruhezustand angeben.

Weitere Informationen finden Sie unter den folgenden Themen:

- Einen manuellen Cluster-Snapshot erstellen
- Wiederherstellung aus einem Cluster-Snapshot
- Kopieren von Amazon DocumentDB-Cluster-Snapshots

- Amazon DocumentDB-Cluster mit aktivierter Speicherverschlüsselung können nicht geändert werden, um die Verschlüsselung zu deaktivieren.
- Alle Instances, automatisierten Backups, Snapshots und Indizes in einem Amazon DocumentDB-Cluster sind mit demselben KMS-Schlüssel verschlüsselt.

Verschlüsseln von Daten während der Übertragung

Sie können Transport Layer Security (TLS) verwenden, um die Verbindung zwischen Ihrer Anwendung und einem Amazon DocumentDB-Cluster zu verschlüsseln. Standardmäßig ist die Verschlüsselung bei der Übertragung für neu erstellte Amazon DocumentDB-Cluster aktiviert. Sie kann bei der Erstellung des Clusters oder zu einem späteren Zeitpunkt optional deaktiviert werden. Wenn die Verschlüsselung während der Übertragung aktiviert ist, sind sichere Verbindungen mit TLS erforderlich, um eine Verbindung mit dem Cluster herzustellen. Weitere Informationen zum Herstellen einer Verbindung zu Amazon DocumentDB über TLS finden Sie unter Programmgesteuertes Herstellen einer Verbindung zu Amazon DocumentDB.

Verwaltung der TLS-Einstellungen für Amazon DocumentDB-Cluster

Die Verschlüsselung bei der Übertragung für einen Amazon DocumentDB-Cluster wird über den TLS-Parameter in einer Cluster-Parametergruppe verwaltet. Sie können Ihre Amazon DocumentDB-Cluster-TLS-Einstellungen mit AWS-Managementkonsole oder AWS Command Line Interface ()AWS CLI verwalten. In den folgenden Abschnitten finden Sie weitere Informationen zum Überprüfen und Ändern Ihrer aktuellen TLS-Einstellungen.

Using the AWS-Managementkonsole

Folgen Sie diesen Schritten, um Verwaltungsaufgaben für die TLS-Verschlüsselung mithilfe der Konsole durchzuführen, z. B. das Identifizieren von Parametergruppen, das Überprüfen des TLS-Werts und das Vornehmen der erforderlichen Änderungen.

> **Note**
>
> Sofern Sie beim Erstellen eines Clusters keine andere Angabe machen, wird Ihr Cluster mit der standardmäßigen Cluster-Parametergruppe erstellt. Die Parameter in der `default` Cluster-Parametergruppe können nicht geändert werden (z. B.). `tls` enabled/disabled Wenn Ihr Cluster also eine `default`-Cluster-Parametergruppe verwendet, müssen Sie den Cluster so ändern, dass er eine nicht standardmäßige Cluster-Parametergruppe verwendet. Zuerst müssen Sie möglicherweise eine benutzerdefinierte Cluster-Parametergruppe erstellen. Weitere Informationen finden Sie unter Erstellen von Amazon DocumentDB-Cluster-Parametergruppen.

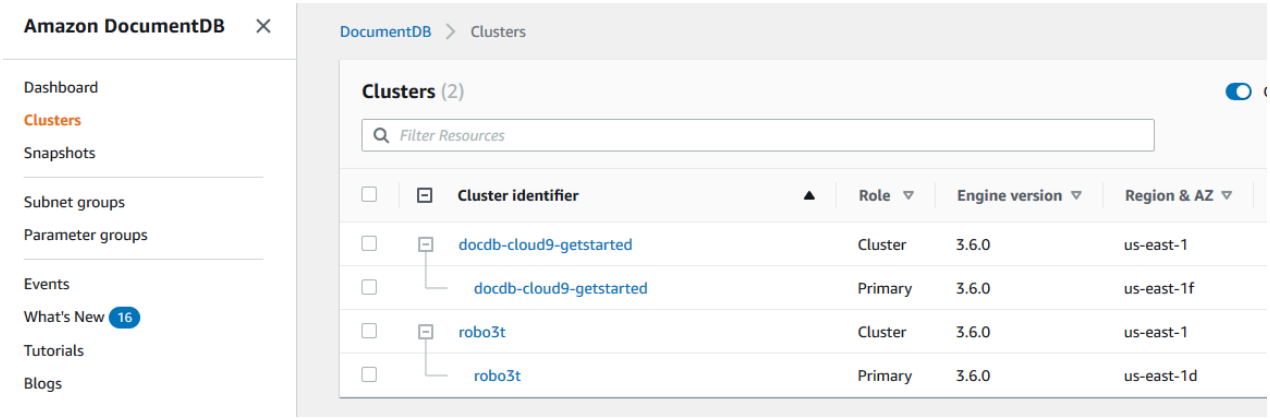
1. Bestimmen Sie, welche Cluster-Parametergruppe Ihr Cluster verwendet.

   a. Öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.

   b. Klicken Sie im Navigationsbereich auf Cluster.

   > **Tip**
   >
   > Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

   c. Beachten Sie, dass im Navigationsfeld Cluster in der Spalte Cluster Identifier sowohl Cluster als auch Instances angezeigt werden. Instanzen sind unter Clustern aufgeführt. Sehen Sie sich den folgenden Screenshot als Referenz an.



   d. Wählen Sie den Cluster aus, an dem Sie interessiert sind.

e. Wählen Sie die Registerkarte „Konfiguration“ und scrollen Sie zum Ende der Cluster-Details und suchen Sie nach der Cluster-Parametergruppe. Der Name der Cluster-Parametergruppe.

   Wenn der Name der Cluster-Parametergruppe `default` lautet (z. B. `default.docdb3.6`), müssen Sie eine benutzerdefinierte Cluster-Parametergruppe erstellen und diese zur Parametergruppe des Clusters machen, bevor Sie fortfahren. Weitere Informationen finden Sie hier:

   1. Erstellen von Amazon DocumentDB-Cluster-Parametergruppen— Wenn Sie keine benutzerdefinierte Cluster-Parametergruppe haben, die Sie verwenden können, erstellen Sie eine.
   2. Ändern eines Amazon DocumentDB-Clusters— Ändern Sie Ihren Cluster so, dass er die benutzerdefinierte Cluster-Parametergruppe verwendet.

2. Bestimmen Sie den aktuellen Wert des **`tls`**-Cluster-Parameters.

   a. Öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.

   b. Wählen Sie im Navigationsbereich Parameter groups (Parametergruppen) aus.

   c. Wählen Sie aus der Liste der Cluster-Parametergruppen den Namen der von Ihnen gewünschten Cluster-Parametergruppe aus.

   d. Suchen Sie den Abschnitt Cluster-Parameter. Suchen Sie in der Liste der Cluster-Parameter die Zeile des `tls`-Cluster-Parameters. An dieser Stelle sind die folgenden vier Spalten wichtig:

      - Name des Cluster-Parameters — Der Name der Cluster-Parameter. Für die Verwaltung von TLS benötigen Sie den `tls`-Cluster-Parameter.
      - Werte — Der aktuelle Wert jedes Clusterparameters.
      - Zulässige Werte — Eine Liste von Werten, die auf einen Clusterparameter angewendet werden können.
      - Typ anwenden — Entweder statisch oder dynamisch. Änderungen an statischen Cluster-Parametern können nur übernommen werden, wenn die Instances neu gestartet werden. Änderungen an dynamischen Cluster-Parametern können entweder sofort übernommen werden oder wenn die Instances neu gestartet werden.

3. Ändern Sie den Wert des **`tls`**-Cluster-Parameters.

Wenn der Wert für `tls` nicht der benötigte Wert ist, ändern Sie ihn für diese Cluster-Parametergruppe. Um den Wert des `tls`-Cluster-Parameters zu ändern, fahren Sie nach dem vorherigen Abschnitt mit folgenden Schritten fort.

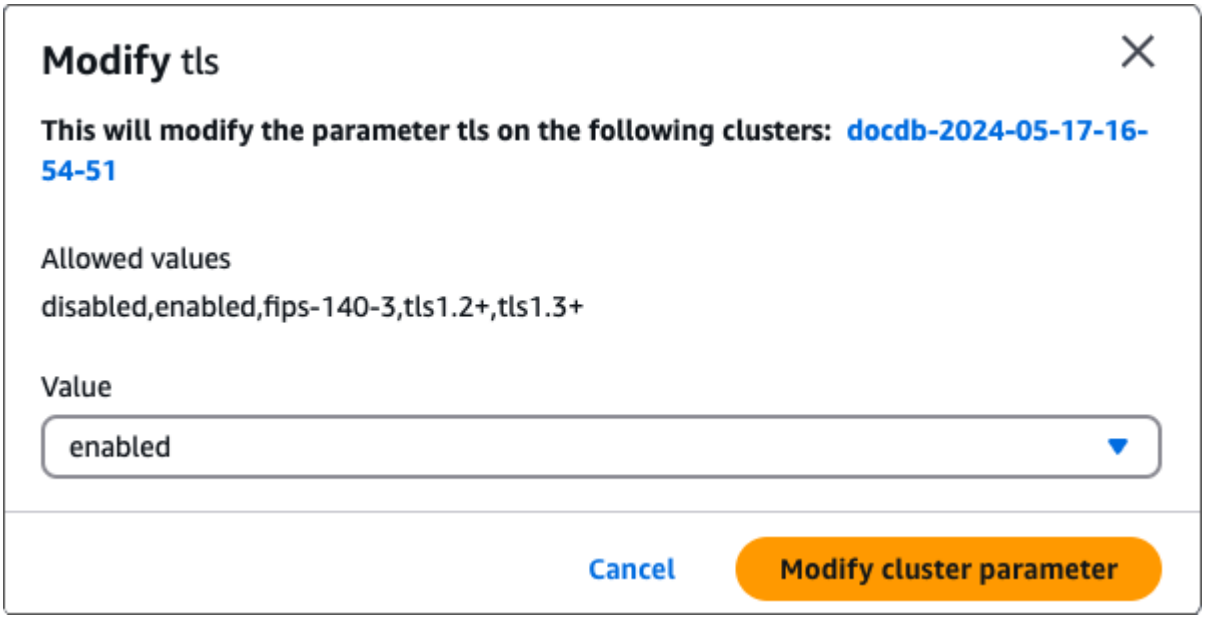
a. Wählen Sie die Schaltfläche links neben dem Namen des Cluster-Parameters (`tls`).

b. Wählen Sie Bearbeiten aus.

c. Um den Wert von zu ändern`tls`, wählen Sie im `tls` Dialogfeld Ändern den gewünschten Wert für den Cluster-Parameter in der Dropdownliste aus.

   Folgende sind gültige Werte:

   - deaktiviert — Deaktiviert TLS
   - aktiviert — Aktiviert TLS. Für Amazon DocumentDB 5.0 und früher aktiviert dies TLS 1.0 bis 1.3, verwenden Sie jedoch TLS 1.2 oder höher. Ab der Nebenversion 5.0.1, Amazon DocumentDB 8.0 und neuer wird auf Amazon DocumentDB 5.0 nur TLS 1.2 und höher unterstützt.
   - fips-140-3 — Aktiviert TLS mit FIPS. Der Cluster akzeptiert nur sichere Verbindungen gemäß den Anforderungen der FIPS-Publikation 140-3 (Federal Information Processing Standards). Dies wird erst ab Amazon DocumentDB 5.0 (Engine-Version 3.0.3727) in den folgenden Regionen unterstützt: ca-central-1, us-west-2, us-east-1, us-east-2, us-gov-east-1, us-gov-west-1.
   - tls1.2+ — Aktiviert TLS Version 1.2 und höher. Dies wird erst ab Amazon DocumentDB 4.0 (Engine-Version 2.0.10980) und Amazon DocumentDB (Engine-Version 3.0.11051) unterstützt.
   - tls1.3+ — Aktiviert TLS Version 1.3 und höher. Dies wird erst ab Amazon DocumentDB 4.0 (Engine-Version 2.0.10980) und Amazon DocumentDB (Engine-Version 3.0.11051) unterstützt.



d. Wählen Sie Modify Cluster Parameter (Cluster-Parameter ändern). Die Änderung wird beim Neustart auf jede Cluster-Instance angewendet.

4. Starten Sie die Amazon DocumentDB-Instance neu.

   Starten Sie jede Instance des Clusters neu, sodass die Änderung für alle Instances im Cluster übernommen wird.

   a. Öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.

   b. Wählen Sie im Navigationsbereich Instances aus.

   c. Um eine Instance anzugeben, die neu gestartet werden soll, suchen Sie die Instance in der Liste der Instances und wählen Sie die Schaltfläche links neben dem Namen aus.

   d. Wählen Sie Actions (Aktionen) und dann Reboot (Neustart) aus. Bestätigen Sie, dass Sie neu starten möchten, indem Sie auf Reboot (Neustart) klicken.

Using the AWS CLI

Folgen Sie diesen Schritten, um Verwaltungsaufgaben für die TLS-Verschlüsselung mithilfe von auszuführen AWS CLI— z. B. das Identifizieren von Parametergruppen, das Überprüfen des TLS-Werts und das Vornehmen der erforderlichen Änderungen.

> **Note**
>
> Sofern Sie beim Erstellen eines Clusters keine andere Angabe machen, wird Ihr Cluster mit der standardmäßigen Cluster-Parametergruppe erstellt. Die Parameter in der `default` Cluster-Parametergruppe können nicht geändert werden (z. B. `tls` enabled/disabled). Wenn Ihr Cluster also eine `default`-Cluster-Parametergruppe verwendet, müssen Sie den Cluster so ändern, dass er eine nicht standardmäßige Cluster-Parametergruppe verwendet. Möglicherweise müssen Sie zuerst eine benutzerdefinierte Cluster-Parametergruppe erstellen. Weitere Informationen finden Sie unter Erstellen von Amazon DocumentDB-Cluster-Parametergruppen.

1. Bestimmen Sie, welche Cluster-Parametergruppe Ihr Cluster verwendet.

   Führen Sie den `describe-db-clusters` Befehl mit den folgenden Optionen aus:

   - `--db-cluster-identifier`
   - `--query`

   Ersetzen Sie im folgenden Beispiel jede *`user input placeholder`* durch die Informationen Ihres Clusters.

   ```
   aws docdb describe-db-clusters \
     --db-cluster-identifier mydocdbcluster \
     --query 'DBClusters[*].[DBClusterIdentifier,DBClusterParameterGroup]'
   ```

   Die Ausgabe dieses Vorgangs sieht ungefähr wie folgt aus (JSON-Format):

   ```
   [
       [
           "mydocdbcluster",
           "myparametergroup"
       ]
   ]
   ```

   Wenn der Name der Cluster-Parametergruppe `default` lautet (z. B. `default.docdb3.6`), benötigen Sie eine benutzerdefinierte Cluster-Parametergruppe und müssen diese zur

Parametergruppe des Clusters machen, bevor Sie fortfahren. Weitere Informationen finden Sie unter den folgenden Themen:

1. Erstellen von Amazon DocumentDB-Cluster-Parametergruppen— Wenn Sie keine benutzerdefinierte Cluster-Parametergruppe haben, die Sie verwenden können, erstellen Sie eine.
2. Ändern eines Amazon DocumentDB-Clusters— Ändern Sie Ihren Cluster so, dass er die benutzerdefinierte Cluster-Parametergruppe verwendet.

2. Bestimmen Sie den aktuellen Wert des **`tls`**-Cluster-Parameters.

   Um weitere Informationen zu dieser Cluster-Parametergruppe zu erhalten, führen Sie den `describe-db-cluster-parameters` Befehl mit den folgenden Optionen aus:

   - `--db-cluster-parameter-group-name`
   - `--query`

     Beschränkt die Ausgabe auf die Interessenfelder: `ParameterNameParameterValue`,`AllowedValues`, und`ApplyType`.

   Ersetzen Sie im folgenden Beispiel jedes *`user input placeholder`* durch die Informationen Ihres Clusters.

```
aws docdb describe-db-cluster-parameters \
  --db-cluster-parameter-group-name myparametergroup \
  --query 'Parameters[*].[ParameterName,ParameterValue,AllowedValues,ApplyType]'
```

   Die Ausgabe dieses Vorgangs sieht ungefähr wie folgt aus (JSON-Format):

```
[
    [
        "audit_logs",
        "disabled",
        "enabled,disabled",
        "dynamic"
    ],
    [
        "tls",
        "disabled",
        "disabled,enabled,fips-140-3,tls1.2+,tls1.3+",
```

```
        "static"
    ],
    [
        "ttl_monitor",
        "enabled",
        "disabled,enabled",
        "dynamic"
    ]
]
```

3. Ändern Sie den Wert des **`tls`**-Cluster-Parameters.

   Wenn der Wert für `tls` nicht der benötigte Wert ist, ändern Sie ihn für diese Cluster-Parametergruppe. Um den Wert des `tls` Cluster-Parameters zu ändern, führen Sie den `modify-db-cluster-parameter-group` Befehl mit den folgenden Optionen aus:

   - `--db-cluster-parameter-group-name` – Erforderlich. Der Name der zu ändernden Cluster-Parametergruppe. Dabei darf es sich nicht um eine `default.*`-Cluster-Parametergruppe handeln.
   - `--parameters` – Erforderlich. Eine Liste der zu ändernden Parameter der Cluster-Parametergruppe.
     - `ParameterName` – Erforderlich. Der Name des zu ändernden Cluster-Parameters.
     - `ParameterValue` – Erforderlich. Der neue Wert für diesen Cluster-Parameter. Muss einer der `AllowedValues` des Cluster-Parameters sein.
       - `enabled`— Der Cluster akzeptiert sichere Verbindungen mit TLS. Für Amazon DocumentDB 5.0 und früher aktiviert dies TLS 1.0 bis 1.3, verwenden Sie jedoch TLS 1.2 oder höher. Ab der Nebenversion 5.0.1, Amazon DocumentDB 8.0 und neuer wird auf Amazon DocumentDB 5.0 nur TLS 1.2 und höher unterstützt.
       - `disabled`— Der Cluster akzeptiert keine sicheren Verbindungen mit TLS.
       - `fips-140-3`— Der Cluster akzeptiert nur sichere Verbindungen gemäß den Anforderungen der FIPS-Publikation 140-3 (Federal Information Processing Standards). Dies wird erst ab Amazon DocumentDB 5.0 (Engine-Version 3.0.3727) in den folgenden Regionen unterstützt: ca-central-1, us-west-2, us-east-1, us-east-2, us-gov-east-1, us-gov-west-1.
       - `tls1.2+`— Der Cluster akzeptiert sichere Verbindungen mit TLS Version 1.2 und höher. Dies wird erst ab Amazon DocumentDB 4.0 (Engine-Version 2.0.10980) und Amazon DocumentDB 5.0 (Engine-Version 3.0.11051) unterstützt.

    - `tls1.3+`— Der Cluster akzeptiert sichere Verbindungen mit TLS Version 1.3 und höher. Dies wird erst ab Amazon DocumentDB 4.0 (Engine-Version 2.0.10980) und Amazon DocumentDB 5.0 (Engine-Version 3.0.11051) unterstützt.
- `ApplyMethod`— Wann diese Änderung angewendet werden soll. Für statische Cluster-Parameter wie `tle` muss dieser Wert `pending-reboot` lauten.
    - `pending-reboot`— Die Änderung wird erst auf eine Instanz angewendet, nachdem sie neu gestartet wurde. Sie müssen jede Cluster-Instance einzeln neu starten, damit die Änderung für alle Instances des Clusters übernommen wird.

Ersetzen Sie in den folgenden Beispielen jedes *`user input placeholder`* durch die Informationen Ihres Clusters.

Der folgende Code wird deaktiviert `tls` und die Änderung wird auf jede Instanz angewendet, wenn sie neu gestartet wird.

```
aws docdb modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=tls,ParameterValue=disabled,ApplyMethod=pending-
reboot"
```

Der folgende Code aktiviert `tls` und wendet die Änderung auf jede Instanz an, wenn sie neu gestartet wird.

```
aws docdb modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=tls,ParameterValue=enabled,ApplyMethod=pending-
reboot"
```

Der folgende Code aktiviert TLS mit `fips-140-3` und wendet die Änderung auf jede Instanz an, wenn sie neu gestartet wird.

```
aws docdb modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup2 \
  --parameters "ParameterName=tls,ParameterValue=fips-140-3,ApplyMethod=pending-
reboot"
```

Die Ausgabe dieser Operation sieht ungefähr wie folgt aus (JSON-Format):

```
{
    "DBClusterParameterGroupName": "myparametergroup"
}
```

4. Starten Sie Ihre Amazon DocumentDB-Instance neu.

   Starten Sie jede Instance des Clusters neu, sodass die Änderung für alle Instances im Cluster übernommen wird. Um eine Amazon DocumentDB-Instance neu zu starten, führen Sie den `reboot-db-instance` Befehl mit der folgenden Option aus:

   - `--db-instance-identifier`

   Der folgende Code startet die Instance `mydocdbinstance` neu.

   Ersetzen Sie in den folgenden Beispielen jedes *`user input placeholder`* durch die Informationen Ihres Clusters.

   Example

   Für Linux, macOS oder Unix:

```
aws docdb reboot-db-instance \
  --db-instance-identifier mydocdbinstance
```

   Für Windows:

```
aws docdb reboot-db-instance ^
  --db-instance-identifier mydocdbinstance
```

   Die Ausgabe dieses Vorgangs sieht ungefähr wie folgt aus (JSON-Format):

```
{
    "DBInstance": {
        "AutoMinorVersionUpgrade": true,
        "PubliclyAccessible": false,
        "PreferredMaintenanceWindow": "fri:09:32-fri:10:02",
        "PendingModifiedValues": {},
        "DBInstanceStatus": "rebooting",
        "DBSubnetGroup": {
```

```
            "Subnets": [
                {
                    "SubnetStatus": "Active",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1a"
                    },
                    "SubnetIdentifier": "subnet-4e26d263"
                },
                {
                    "SubnetStatus": "Active",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1c"
                    },
                    "SubnetIdentifier": "subnet-afc329f4"
                },
                {
                    "SubnetStatus": "Active",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1e"
                    },
                    "SubnetIdentifier": "subnet-b3806e8f"
                },
                {
                    "SubnetStatus": "Active",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1d"
                    },
                    "SubnetIdentifier": "subnet-53ab3636"
                },
                {
                    "SubnetStatus": "Active",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1b"
                    },
                    "SubnetIdentifier": "subnet-991cb8d0"
                },
                {
                    "SubnetStatus": "Active",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1f"
                    },
                    "SubnetIdentifier": "subnet-29ab1025"
                }
            ],
```

```
            "SubnetGroupStatus": "Complete",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-91280df6",
            "DBSubnetGroupName": "default"
        },
        "PromotionTier": 2,
        "DBInstanceClass": "db.r5.4xlarge",
        "InstanceCreateTime": "2018-11-05T23:10:49.905Z",
        "PreferredBackupWindow": "00:00-00:30",
        "KmsKeyId": "arn:aws:kms:us-east-1:012345678901:key/0961325d-a50b-44d4-
b6a0-a177d5ff730b",
        "StorageEncrypted": true,
        "VpcSecurityGroups": [
            {
                "Status": "active",
                "VpcSecurityGroupId": "sg-77186e0d"
            }
        ],
        "EngineVersion": "3.6.0",
        "DbiResourceId": "db-SAMPLERESOURCEID",
        "DBInstanceIdentifier": "mydocdbinstance",
        "Engine": "docdb",
        "AvailabilityZone": "us-east-1a",
        "DBInstanceArn": "arn:aws:rds:us-east-1:012345678901:db:sample-cluster-
instance-00",
        "BackupRetentionPeriod": 1,
        "Endpoint": {
            "Address": "mydocdbinstance.corcjozrlsfc.us-
east-1.docdb.amazonaws.com",
            "Port": 27017,
            "HostedZoneId": "Z2R2ITUGPM61AM"
        },
        "DBClusterIdentifier": "mydocdbcluster"
    }
}
```


Es dauert einige Minuten, bis Ihre Instance neu gestartet wird. Sie können die Instance nur verwenden, wenn ihr Status available ist. Sie können mit der Konsole oder der AWS CLI den Status der Instance überwachen. Weitere Informationen finden Sie unter Überwachung des Status einer Amazon DocumentDB-Instance.

Schlüsselverwaltung

Amazon DocumentDB verwendet AWS Key Management Service (AWS KMS), um Verschlüsselungsschlüssel abzurufen und zu verwalten. AWS KMS kombiniert sichere, hochverfügbare Hard- und Software zu einem für die Cloud skalierten Schlüsselverwaltungssystem. Mithilfe AWS KMS können Sie Verschlüsselungsschlüssel erstellen und die Richtlinien definieren, die steuern, wie diese Schlüssel verwendet werden können. AWS KMS unterstützt AWS CloudTrail, sodass Sie die Verwendung von Schlüsseln überprüfen können, um sicherzustellen, dass die Schlüssel ordnungsgemäß verwendet werden.

Sie können Ihre KMS-Schlüssel in Kombination mit Amazon DocumentDB und unterstützten AWS Diensten wie Amazon Simple Storage Service (Amazon S3), Amazon Relational Database Service (Amazon RDS), Amazon Elastic Block Store (Amazon EBS) und Amazon Redshift verwenden. Eine Liste der Dienste, die dies unterstützen AWS KMS, finden Sie AWS KMS im Entwicklerhandbuch unter So nutzen Sie AWS Dienste. AWS Key Management Service Weitere Informationen finden AWS KMS Sie unter Was ist AWS Key Management Service?

Identitäts- und Zugriffsmanagement für Amazon DocumentDB

AWS Identity and Access Management (IAM) ist ein Programm AWS-Service , das einem Administrator hilft, den Zugriff auf Ressourcen sicher zu kontrollieren. AWS IAM-Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um Amazon DocumentDB-Ressourcen zu nutzen. IAM ist eine AWS-Service , die Sie ohne zusätzliche Kosten verwenden können.

Themen

- Zielgruppe
- Authentifizierung mit Identitäten
- Verwalten des Zugriffs mit Richtlinien
- Wie Amazon DocumentDB mit IAM funktioniert
- Identity-based Beispiele für Richtlinien für Amazon DocumentDB
- Fehlerbehebung bei Identität und Zugriff auf Amazon DocumentDB
- Verwaltung der Zugriffsberechtigungen für Ihre Amazon DocumentDB-Ressourcen

- Verwendung identitätsbasierter Richtlinien (IAM-Richtlinien) für Amazon DocumentDB
- AWS verwaltete Richtlinien für Amazon DocumentDB
- Amazon DocumentDB-API-Berechtigungen: Referenz zu Aktionen, Ressourcen und Bedingungen

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von Ihrer Rolle ab:

- Servicebenutzer – Fordern Sie von Ihrem Administrator Berechtigungen an, wenn Sie nicht auf Features zugreifen können (siehe Fehlerbehebung bei Identität und Zugriff auf Amazon DocumentDB).
- Serviceadministrator – Bestimmen Sie den Benutzerzugriff und stellen Sie Berechtigungsanfragen (siehe Wie Amazon DocumentDB mit IAM funktioniert).
- IAM-Administrator – Schreiben Sie Richtlinien zur Zugriffsverwaltung (siehe Identity-based Beispiele für Richtlinien für Amazon DocumentDB).

Authentifizierung mit Identitäten

Bei der Authentifizierung melden Sie sich AWS mit Ihren Identitätsdaten an. Sie müssen als IAM-Benutzer oder als IAM-Benutzer authentifiziert sein, indem Sie eine IAM-Rolle übernehmen. Root-Benutzer des AWS-Kontos

Sie können sich mit Anmeldeinformationen aus einer Identitätsquelle wie AWS IAM Identity Center (IAM Identity Center), Single Sign-On-Authentifizierung oder Anmeldeinformationen als Verbundidentität anmelden. Google/Facebook Weitere Informationen zum Anmelden finden Sie unter So melden Sie sich bei Ihrem AWS-Konto an im Benutzerhandbuch für AWS-Anmeldung .

AWS Stellt für den programmatischen Zugriff ein SDK und eine CLI zum kryptografischen Signieren von Anfragen bereit. Weitere Informationen finden Sie unter AWS Signature Version 4 for API requests im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, dem AWS-Konto Root-Benutzer, der vollständigen Zugriff auf alle AWS-Services Ressourcen hat. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Eine Liste der

Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Tasks that require root user credentials im IAM-Benutzerhandbuch.

Verbundidentität

Es hat sich bewährt, menschliche Benutzer dazu zu verpflichten, einen Verbund mit einem Identitätsanbieter zu verwenden, um AWS-Services mithilfe temporärer Anmeldeinformationen darauf zuzugreifen.

Bei einer Verbundidentität handelt es sich um einen Benutzer aus Ihrem Unternehmensverzeichnis, einem Web-Identitätsanbieter oder Directory Service einem Benutzer, der AWS-Services mithilfe von Anmeldeinformationen aus einer Identitätsquelle zugreift. Verbundene Identitäten übernehmen Rollen, die temporäre Anmeldeinformationen bereitstellen.

Für die zentrale Zugriffsverwaltung empfehlen wir AWS IAM Identity Center. Weitere Informationen finden Sie unter Was ist IAM Identity Center? im AWS IAM Identity Center -Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität mit bestimmten Berechtigungen für eine einzelne Person oder Anwendung. Wir empfehlen die Verwendung temporärer Anmeldeinformationen anstelle von IAM-Benutzern mit langfristigen Anmeldeinformationen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Erfordern, dass menschliche Benutzer einen Verbund mit einem Identitätsanbieter verwenden, um AWS mithilfe temporärer Anmeldeinformationen auf den Zugriff zuzugreifen.

Eine IAM-Gruppe spezifiziert eine Sammlung von IAM-Benutzern und erleichtert die Verwaltung von Berechtigungen für große Gruppen von Benutzern. Weitere Informationen finden Sie unter Anwendungsfälle für IAM-Benutzer im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität mit spezifischen Berechtigungen, die temporäre Anmeldeinformationen bereitstellt. Sie können eine Rolle übernehmen, indem Sie von einer Benutzer- zu einer IAM-Rolle (Konsole) wechseln AWS CLI oder eine AWS API-Operation aufrufen. Weitere Informationen finden Sie unter Methoden, um eine Rolle zu übernehmen im IAM-Benutzerhandbuch.

IAM-Rollen sind nützlich für den Verbundbenutzer-Zugriff, temporäre IAM-Benutzerberechtigungen, kontoübergreifenden Zugriff, serviceübergreifenden Zugriff und Anwendungen, die auf Amazon EC2

laufen. Weitere Informationen finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Sie steuern den Zugriff, AWS indem Sie Richtlinien erstellen und sie an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie definiert Berechtigungen, wenn sie einer Identität oder Ressource zugeordnet sind. AWS wertet diese Richtlinien aus, wenn ein Principal eine Anfrage stellt. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Mit Hilfe von Richtlinien legen Administratoren fest, wer Zugriff auf was hat, indem sie definieren, welches Prinzipal welche Aktionen auf welchen Ressourcenund unter welchen Bedingungendurchführen darf.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator erstellt IAM-Richtlinien und fügt sie zu Rollen hinzu, die die Benutzer dann übernehmen können. IAM-Richtlinien definieren Berechtigungen unabhängig von der Methode, die zur Ausführung der Operation verwendet wird.

Identity-based Richtlinien

Identity-based Richtlinien sind Richtliniendokumente für JSON-Berechtigungen, die Sie an eine Identität (Benutzer, Gruppe oder Rolle) anhängen. Diese Richtlinien steuern, welche Aktionen Identitäten für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Identity-based Richtlinien können integrierte Richtlinien (direkt in eine einzelne Identität eingebettet) oder verwaltete Richtlinien (eigenständige Richtlinien, die mehreren Identitäten zugeordnet sind) sein. Informationen dazu, wie Sie zwischen verwalteten und Inline-Richtlinien wählen, finden Sie unter Choose between managed policies and inline policies im IAM-Benutzerhandbuch.

Resource-based Richtlinien

Resource-based Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anhängen. Beispiele hierfür sind Vertrauensrichtlinien für IAM-Rollen und Amazon S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren

sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben.

Resource-based Richtlinien sind Inline-Richtlinien, die sich in diesem Dienst befinden. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Weitere Richtlinientypen

AWS unterstützt zusätzliche Richtlinientypen, mit denen die maximale Anzahl von Berechtigungen festgelegt werden kann, die von gängigeren Richtlinientypen gewährt werden:

- Berechtigungsgrenzen – Eine Berechtigungsgrenze legt die maximalen Berechtigungen fest, die eine identitätsbasierte Richtlinie einer IAM-Entität erteilen kann. Weitere Informationen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im -IAM-Benutzerhandbuch.
- Service-Kontrollrichtlinien (SCPs) – SCPs legen die maximalen Berechtigungen für eine Organisation oder Organisationseinheit in AWS Organizations fest. Weitere Informationen finden Sie unter Service-Kontrollrichtlinien im AWS Organizations -Benutzerhandbuch.
- Ressourcen-Kontrollrichtlinien (RCPs) – RCPs definieren die maximale Anzahl an Berechtigungen, die Ressourcen in Ihren Konten zur Verfügung stehen. Weitere Informationen finden Sie unter Ressourcen-Kontrollrichtlinien im AWS Organizations -Benutzerhandbuch.
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die als Parameter übergeben werden, wenn Sie eine temporäre Sitzung für eine Rolle oder einen Verbundbenutzer erstellen. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn mehrere Arten von Richtlinien für eine Anfrage gelten, sind die daraus resultierenden Berechtigungen komplizierter zu verstehen. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie im IAM-Benutzerhandbuch unter Logik zur Richtlinienauswertung.

Wie Amazon DocumentDB mit IAM funktioniert

Bevor Sie IAM verwenden, um den Zugriff auf Amazon DocumentDB zu verwalten, sollten Sie sich mit den IAM-Funktionen vertraut machen, die Sie mit Amazon DocumentDB verwenden können.

IAM-Funktionen, die Sie mit Amazon DocumentDB verwenden können

IAM-Feature	Instance-based Cluster	Elastische Cluster
Identity-based Richtlinien	Ja	Ja
Resource-based Richtlinien	Nein	Nein
Richtlinienaktionen	Ja	Ja
Richtlinienressourcen	Ja	Ja
Richtlinienbedingungsschlüssel (servicespezifisch)	Ja	Ja
ACLs	Nein	Nein
ABAC (Tags in Richtlinien)	Teilweise	Ja
Temporäre Anmeldeinformationen	Ja	Ja
Prinzipalberechtigungen	Ja	Ja
Servicerollen	Ja	Ja
Service-linked Rollen	Nein	Ja

Einen allgemeinen Überblick darüber, wie Amazon DocumentDB und andere AWS Dienste mit den meisten IAM-Funktionen funktionieren, finden Sie im IAM-Benutzerhandbuch unter AWS Services, die mit IAM funktionieren.

Identity-based Richtlinien für Amazon DocumentDB

Unterstützt Richtlinien auf Identitätsbasis: Ja

Identity-based Richtlinien sind Richtliniendokumente für JSON-Berechtigungen, die Sie an eine Identität anhängen können, z. B. an einen IAM-Benutzer, eine Benutzergruppe oder eine Rolle. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien

finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Identity-based Beispiele für Richtlinien für Amazon DocumentDB

Beispiele für identitätsbasierte Richtlinien von Amazon DocumentDB finden Sie unter. Identity-based Beispiele für Richtlinien für Amazon DocumentDB

Resource-based Richtlinien in Amazon DocumentDB

Unterstützt ressourcenbasierte Richtlinien: Nein

Resource-based Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anhängen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Principals können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Um kontoübergreifenden Zugriff zu ermöglichen, können Sie ein gesamtes Konto oder IAM-Entitäten in einem anderen Konto als Prinzipal in einer ressourcenbasierten Richtlinie angeben. Weitere Informationen finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Politische Maßnahmen für Amazon DocumentDB

Unterstützt Richtlinienaktionen: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Nehmen Sie Aktionen in eine Richtlinie auf, um Berechtigungen zur Ausführung des zugehörigen Vorgangs zu gewähren.

> **Note**
>
> Für bestimmte Verwaltungsfunktionen verwendet Amazon DocumentDB eine Betriebstechnologie, die mit Amazon Relational Database Service (Amazon RDS) gemeinsam genutzt wird.
> Eine Liste der RDS-Aktionen finden Sie unter Von Amazon Relational Database Service definierte Aktionen in der Service Authorization Reference.
> Informationen zu den Richtlinienaktionen für Amazon DocumentDB Elastic Clusters finden Sie unter Von Amazon DocumentDB Elastic Clusters definierte Aktionen in der Service Authorization Reference.

Richtlinienaktionen in Amazon DocumentDB verwenden vor der Aktion das folgende Präfix:

```
aws
```

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie mit Kommata:

```
"Action": [
      "aws:action1",
      "aws:action2"
         ]
```

Beispiele für identitätsbasierte Richtlinien von Amazon DocumentDB finden Sie unter. Identity-based Beispiele für Richtlinien für Amazon DocumentDB

Richtlinienressourcen für Amazon DocumentDB

Unterstützt Richtlinienressourcen: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Als Best Practice geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*"
```

> **Note**
>
> Für bestimmte Verwaltungsfunktionen verwendet Amazon DocumentDB eine Betriebstechnologie, die mit Amazon Relational Database Service (Amazon RDS) gemeinsam genutzt wird.
> Eine Liste der RDS-Ressourcentypen und ihrer ARNs finden Sie unter Von Amazon Relational Database Service definierte Ressourcen in der Service Authorization Reference. Weitere Informationen dazu, mit welchen Aktionen Sie den ARN jeder Ressource angeben können, finden Sie unter Von Amazon Relational Database Service definierte Aktionen.
> Informationen zu den Ressourcentypen für Amazon DocumentDB Elastic Clusters finden Sie in der Service Authorization Reference unter Von Amazon DocumentDB Elastic Clusters definierte Ressourcentypen.

Beispiele für identitätsbasierte Richtlinien von Amazon DocumentDB finden Sie unter. Identity-based Beispiele für Richtlinien für Amazon DocumentDB

Bedingungsschlüssel für Richtlinien für Amazon DocumentDB

Unterstützt servicespezifische Richtlinienbedingungsschlüssel: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Condition` gibt an, wann Anweisungen auf der Grundlage definierter Kriterien ausgeführt werden. Sie können bedingte Ausdrücke erstellen, die Bedingungsoperatoren verwenden, z. B. ist gleich oder kleiner als, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt. Alle AWS globalen Bedingungsschlüssel finden Sie unter AWS Globale Bedingungskontextschlüssel im IAM-Benutzerhandbuch.

> **Note**
>
> Für bestimmte Verwaltungsfunktionen verwendet Amazon DocumentDB eine Betriebstechnologie, die mit Amazon Relational Database Service (Amazon RDS) gemeinsam genutzt wird.
> Eine Liste der RDS-Bedingungsschlüssel finden Sie unter Bedingungsschlüssel für Amazon Relational Database Service in der Service Authorization Reference. Informationen zu den Aktionen und Ressourcen, mit denen Sie einen Bedingungsschlüssel verwenden können, finden Sie unter Von Amazon Relational Database Service definierte Aktionen.
> Informationen zu Bedingungsschlüsseln für Amazon DocumentDB Elastic Clusters finden Sie unter Bedingungsschlüssel für Amazon DocumentDB Elastic Clusters in der Service Authorization Reference.

Beispiele für identitätsbasierte Richtlinien von Amazon DocumentDB finden Sie unter. Identity-based Beispiele für Richtlinien für Amazon DocumentDB

ACLs in Amazon DocumentDB

Unterstützt ACLs: Nein

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

ABAC mit Amazon DocumentDB

> **Note**
>
> ABAC wird für instanzbasierte Cluster nur teilweise unterstützt, für elastische Cluster jedoch.

Attribute-based Access Control (ABAC) ist eine Autorisierungsstrategie, die Berechtigungen auf der Grundlage von Attributen definiert, die als Tags bezeichnet werden. Sie können Tags an IAM-Entitäten und AWS -Ressourcen anhängen und dann ABAC-Richtlinien so entwerfen, dass Operationen möglich sind, wenn das Tag des Principals mit dem Tag auf der Ressource übereinstimmt.

Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungselement einer Richtlinie Tag-Informationen an, indem Sie die Schlüssel `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, oder Bedingung `aws:TagKeys` verwenden.

Wenn ein Service alle drei Bedingungsschlüssel für jeden Ressourcentyp unterstützt, lautet der Wert für den Service Ja. Wenn ein Service alle drei Bedingungsschlüssel für nur einige Ressourcentypen unterstützt, lautet der Wert Teilweise.

Weitere Informationen zu ABAC finden Sie unter Definieren von Berechtigungen mit ABAC-Autorisierung im IAM-Benutzerhandbuch. Um ein Tutorial mit Schritten zur Einstellung von ABAC anzuzeigen, siehe Attributbasierte Zugriffskontrolle (ABAC) verwenden im IAM-Benutzerhandbuch.

Verwenden temporärer Anmeldeinformationen mit Amazon DocumentDB

Unterstützt temporäre Anmeldeinformationen: Ja

Temporäre Anmeldeinformationen ermöglichen den kurzfristigen Zugriff auf AWS Ressourcen und werden automatisch erstellt, wenn Sie den Verbund verwenden oder die Rollen wechseln. AWS empfiehlt, temporäre Anmeldeinformationen dynamisch zu generieren, anstatt langfristige Zugriffsschlüssel zu verwenden. Weitere Informationen finden Sie unter Temporäre Anmeldeinformationen in IAM und AWS-Services , die mit IAM funktionieren im IAM-Benutzerhandbuch.

Cross-service Hauptberechtigungen für Amazon DocumentDB

Unterstützt Forward Access Sessions (FAS): Ja

Forward Access Sessions (FAS) verwenden die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, in Kombination mit der Anforderung, Anfragen an nachgelagerte Dienste AWS-Service zu stellen. Einzelheiten zu den Richtlinien für FAS-Anforderungen finden Sie unter Zugriffssitzungen weiterleiten.

Servicerollen für Amazon DocumentDB

Unterstützt Servicerollen: Ja

Eine Servicerolle ist eine IAM-Rolle, die ein Service annimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.

> **Warning**
>
> Das Ändern der Berechtigungen für eine Servicerolle kann die Amazon DocumentDB-Funktionalität beeinträchtigen. Bearbeiten Sie Servicerollen nur, wenn Amazon DocumentDB dazu eine Anleitung gibt.

Service-linked Rollen für Amazon DocumentDB

> **Note**
>
> Service-linked Rollen werden für instanzbasierte Cluster nicht unterstützt, für elastische Cluster jedoch unterstützt.

Eine dienstverknüpfte Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Dienst kann die Rolle übernehmen, eine Aktion in Ihrem Namen auszuführen. Service-linked Rollen werden in Ihrem angezeigt AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

Details zum Erstellen oder Verwalten von serviceverknüpften Rollen finden Sie unter AWS -Services, die mit IAM funktionieren. Suchen Sie einen Dienst in der Tabelle, die eine `Yes` in der Service-linked Rollenspalte enthält. Wählen Sie den Link Yes (Ja) aus, um die Dokumentation für die serviceverknüpfte Rolle für diesen Service anzuzeigen.

Identity-based Beispiele für Richtlinien für Amazon DocumentDB

Standardmäßig sind Benutzer und Rollen nicht berechtigt, Amazon DocumentDB-Ressourcen zu erstellen oder zu ändern. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter Erstellen von IAM-Richtlinien (Konsole) im IAM-Benutzerhandbuch.

Einzelheiten zu den von Amazon DocumentDB definierten Aktionen und Ressourcentypen, einschließlich des Formats der ARNs für jeden der Ressourcentypen, finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon Relational Database Service in der Service Authorization Reference.

> **Note**
>
> Für bestimmte Verwaltungsfunktionen verwendet Amazon DocumentDB eine Betriebstechnologie, die mit Amazon Relational Database Service (Amazon RDS) gemeinsam genutzt wird.
> Informationen zu Richtlinienaktionen für elastische Amazon DocumentDB-Cluster finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon DocumentDB Elastic Clusters in der Service Authorization Reference.

Themen

- Best Practices für Richtlinien
- Verwenden der Amazon DocumentDB-Konsole
- Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

Best Practices für Richtlinien

Identity-based Richtlinien legen fest, ob jemand Amazon DocumentDB-Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Wenn Sie identitätsbasierte Richtlinien erstellen oder bearbeiten, folgen Sie diesen Richtlinien und Empfehlungen:

- Beginnen Sie mit AWS verwalteten Richtlinien und wechseln Sie zu Berechtigungen mit geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um Ihren Benutzern und Workloads zunächst Berechtigungen zu gewähren. Sie sind in Ihrem verfügbar. AWS-Konto Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie unter Von AWS verwaltete Richtlinien oder Von AWS verwaltete Richtlinien für Auftragsfunktionen im IAM-Benutzerhandbuch.
- Anwendung von Berechtigungen mit den geringsten Rechten – Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum

Anwenden von Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.

- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese im Rahmen einer bestimmten Aktion verwendet werden AWS-Service, z. CloudFormation B. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.
- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter Richtlinienvalidierung mit IAM Access Analyzer im IAM-Benutzerhandbuch.
- Mehrstufige Authentifizierung (MFA) erforderlich — Wenn Sie ein Szenario haben, in dem IAM-Benutzer oder ein Root-Benutzer in Ihrem System erforderlich sind AWS-Konto, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter Sicherer API-Zugriff mit MFA im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Best Practices für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Verwenden der Amazon DocumentDB-Konsole

Für den Zugriff auf die Amazon DocumentDB-Konsole (mit MongoDB-Kompatibilität) benötigen Sie ein Mindestmaß an Berechtigungen. Diese Berechtigungen müssen es Ihnen ermöglichen, Details zu den Amazon DocumentDB-Ressourcen in Ihrem aufzulisten und anzuzeigen. AWS-Konto Wenn Sie eine identitätsbasierte Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Entitäten (Benutzer oder Rollen) mit dieser Richtlinie.

Sie müssen Benutzern, die nur die AWS CLI oder die AWS API aufrufen, keine Mindestberechtigungen für die Konsole gewähren. Stattdessen sollten Sie nur Zugriff auf die Aktionen zulassen, die der API-Operation entsprechen, die die Benutzer ausführen möchten.

Um sicherzustellen, dass Benutzer und Rollen die Amazon DocumentDB-Konsole weiterhin verwenden können, fügen Sie den Entitäten auch die Amazon DocumentDB-Richtlinie *ConsoleAccess* oder die *ReadOnly* AWS verwaltete Richtlinie hinzu. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zu einem Benutzer im IAM-Benutzerhandbuch.

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie beinhaltet Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder programmgesteuert mithilfe der OR-API. AWS CLI AWS

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
```

```
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Fehlerbehebung bei Identität und Zugriff auf Amazon DocumentDB

Verwenden Sie die folgenden Informationen, um häufig auftretende Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit Amazon DocumentDB und IAM auftreten können.

Themen

- Ich bin nicht berechtigt, eine Aktion in Amazon DocumentDB auszuführen
- Ich bin nicht berechtigt, iam auszuführen: PassRole
- Ich möchte Leute außerhalb von mir zulassen AWS-Konto um auf meine Amazon DocumentDB-Ressourcen zuzugreifen

Ich bin nicht berechtigt, eine Aktion in Amazon DocumentDB auszuführen

Wenn Sie eine Fehlermeldung erhalten, dass Sie nicht zur Durchführung einer Aktion berechtigt sind, müssen Ihre Richtlinien aktualisiert werden, damit Sie die Aktion durchführen können.

Der folgende Beispielfehler tritt auf, wenn der IAM-Benutzer `mateojackson` versucht, über die Konsole Details zu einer fiktiven *`my-example-widget`*-Ressource anzuzeigen, jedoch nicht über `aws:`*`GetWidget`*-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 aws:GetWidget on resource: my-example-widget
```

In diesem Fall muss die Richtlinie für den Benutzer `mateojackson` aktualisiert werden, damit er mit der `aws:`*`GetWidget`*-Aktion auf die *`my-example-widget`*-Ressource zugreifen kann.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich bin nicht berechtigt, iam auszuführen: PassRole

Wenn Sie eine Fehlermeldung erhalten, dass Sie nicht berechtigt sind, die `iam:PassRole` Aktion auszuführen, müssen Ihre Richtlinien aktualisiert werden, damit Sie eine Rolle an Amazon DocumentDB übergeben können.

In einigen AWS-Services Fällen können Sie eine vorhandene Rolle an diesen Service übergeben, anstatt eine neue Servicerolle oder eine serviceverknüpfte Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in Amazon DocumentDB auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Leute außerhalb von mir zulassen AWS-Konto um auf meine Amazon DocumentDB-Ressourcen zuzugreifen

Sie können eine Rolle erstellen, mit der Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation auf Ihre Ressourcen zugreifen können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Im Fall von Diensten, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (Access Control Lists, ACLs) verwenden, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen darüber, ob Amazon DocumentDB diese Funktionen unterstützt, finden Sie unterWie Amazon DocumentDB mit IAM funktioniert.
- Weitere Informationen zum Bereitstellen des Zugriffs auf Ihre Ressourcen in Ihrem eigenen AWS-Konten Bereich finden Sie im IAM-Benutzerhandbuch unter Bereitstellen des Zugriffs für einen IAM-Benutzer in einem anderen Benutzer AWS-Konto , dessen Eigentümer Sie sind.

- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren AWS-Konten, finden Sie im IAM-Benutzerhandbuch unter Gewähren des Zugriffs auf Ressourcen, die Dritten AWS-Konten gehören.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Verwaltung der Zugriffsberechtigungen für Ihre Amazon DocumentDB-Ressourcen

Jede AWS Ressource gehört einem AWS-Konto, und die Berechtigungen zum Erstellen oder Zugreifen auf die Ressourcen werden durch Berechtigungsrichtlinien geregelt. Ein Kontoadministrator kann IAM-Identitäten (d. h. Benutzern, Gruppen und Rollen) Berechtigungsrichtlinien zuordnen, und einige Dienste (z. B. AWS Lambda) unterstützen auch das Anhängen von Berechtigungsrichtlinien an Ressourcen.

> **Note**
>
> Ein Kontoadministrator (oder Administratorbenutzer) ist ein Benutzer mit Administratorberechtigungen. Weitere Informationen finden Sie unter Bewährte Methoden für IAM im IAM-Benutzerhandbuch.

Themen

- Ressourcen und Abläufe von Amazon DocumentDB
- Grundlegendes zum Eigentum an Ressourcen
- Verwaltung des Zugriffs auf -Ressourcen
- Spezifizierung von Richtlinienelementen: Aktionen, Auswirkungen, Ressourcen und Prinzipien
- Angeben von Bedingungen in einer Richtlinie

Ressourcen und Abläufe von Amazon DocumentDB

In Amazon DocumentDB ist die primäre Ressource ein Cluster. Amazon DocumentDB unterstützt andere Ressourcen, die zusammen mit der primären Ressource verwendet werden können, wie Instances, Parametergruppen und Veranstaltungsabonnements. Diese Ressourcen werden als Unterressourcen bezeichnet.

Diese Ressourcen und Unterressourcen sind eindeutigen Amazon-Ressourcennamen (ARNs) zugeordnet (siehe Tabelle unten).

Ressourcentyp	ARN-Format
Cluster	`arn:aws:rds:` *`region`*`:`*`account-id`* `:cluster:` *`db-cluster-name`*
Cluster-Parametergruppe	`arn:aws:rds:` *`region`*`:`*`account-id`* `:cluster-pg:` *`cluster-parameter-group-name`*
Cluster-Snapshot	`arn:aws:rds:` *`region`*`:`*`account-id`* `:cluster-snapshot:` *`cluster-snapshot-name`*
Instance	`arn:aws:rds:` *`region`*`:`*`account-id`* `:db:`*`db-instance-name`*
Sicherheitsgruppe	`arn:aws:rds:` *`region`*`:`*`account-id`* `:secgrp:`*`security-group-name`*
Subnetzgruppe	`arn:aws:rds:` *`region`*`:`*`account-id`* `:subgrp:`*`subnet-group-name`*

Amazon DocumentDB bietet eine Reihe von Vorgängen für die Arbeit mit den Amazon DocumentDB-Ressourcen. Eine Liste der verfügbaren Operationen finden Sie unter Aktionen.

Grundlegendes zum Eigentum an Ressourcen

Ein Ressourcenbesitzer ist derjenige AWS-Konto , der eine Ressource erstellt hat. Das heißt, der Ressourcenbesitzer ist derjenige AWS-Konto der Prinzipalentität (das Stammkonto, ein IAM-Benutzer oder eine IAM-Rolle), die die Anforderung authentifiziert, mit der die Ressource erstellt wird. Die Funktionsweise wird anhand der folgenden Beispiele deutlich:

- Wenn Sie Ihre Root-Kontoanmeldedaten verwenden, AWS-Konto um eine Amazon DocumentDB-Ressource, z. B. eine Instance, zu erstellen, sind Sie AWS-Konto der Eigentümer der Amazon DocumentDB-Ressource.
- Wenn Sie einen IAM-Benutzer in Ihrem Konto erstellen AWS-Konto und diesem Benutzer Berechtigungen zum Erstellen von Amazon DocumentDB-Ressourcen gewähren, kann der Benutzer Amazon DocumentDB-Ressourcen erstellen. Ihr Unternehmen AWS-Konto, zu dem der Benutzer gehört, besitzt jedoch die Amazon DocumentDB-Ressourcen.
- Wenn Sie in Ihrem Konto eine IAM-Rolle AWS-Konto mit Berechtigungen zum Erstellen von Amazon DocumentDB-Ressourcen erstellen, kann jeder, der diese Rolle übernehmen kann, Amazon DocumentDB-Ressourcen erstellen. Ihr Unternehmen AWS-Konto, zu dem die Rolle gehört, besitzt die Amazon DocumentDB-Ressourcen.

Verwaltung des Zugriffs auf -Ressourcen

Eine Berechtigungsrichtlinie beschreibt, wer Zugriff auf welche Objekte hat. Im folgenden Abschnitt werden die verfügbaren Optionen zum Erstellen von Berechtigungsrichtlinien erläutert.

> **Note**
>
> In diesem Abschnitt wird die Verwendung von IAM im Kontext von Amazon DocumentDB beschrieben. Er enthält keine detaillierten Informationen über den IAM-Service. Eine umfassende IAM-Dokumentation finden Sie unter Was ist IAM? im IAM-Benutzerhandbuch. Für Informationen über die Syntax und Beschreibungen von AWS -IAM-Richtlinien lesen Sie die IAM-Richtlinienreferenz im IAM-Benutzerhandbuch.

An eine IAM-Identität angefügte Richtlinien werden als identitätsbasierte Richtlinien (IAM-Richtlinien) bezeichnet. An Ressourcen angehängte Richtlinien werden als ressourcenbasierte Richtlinien bezeichnet. Amazon DocumentDB unterstützt nur identitätsbasierte Richtlinien (IAM-Richtlinien).

Themen

- Identity-based Richtlinien (IAM-Richtlinien)
- Resource-based Richtlinien

Identity-based Richtlinien (IAM-Richtlinien)

Richtlinien können IAM-Identitäten angefügt werden. Sie können z. B. Folgendes tun:

- Hängen Sie eine Berechtigungsrichtlinie an einen Benutzer oder eine Gruppe in Ihrem Konto an — Ein Kontoadministrator kann eine Berechtigungsrichtlinie verwenden, die einem bestimmten Benutzer zugeordnet ist, um diesem Benutzer die Erlaubnis zu erteilen, eine Amazon DocumentDB-Ressource, z. B. eine Instance, zu erstellen.
- Einer Rolle eine Berechtigungsrichtlinie zuweisen (kontoübergreifende Berechtigungen gewähren) – Sie können einer IAM-Rolle eine identitätsbasierte Berechtigungsrichtlinie zuweisen, um kontoübergreifende Berechtigungen zu erteilen. Ein Administrator kann beispielsweise wie folgt eine Rolle erstellen, um einem anderen Konto AWS-Konto oder einem AWS Dienst kontoübergreifende Berechtigungen zu gewähren:
  1. Der Administrator von Konto A erstellt eine IAM-Rolle und fügt ihr eine Berechtigungsrichtlinie an, die Berechtigungen für Ressourcen in Konto A erteilt.
  2. Der Administrator von Konto A weist der Rolle eine Vertrauensrichtlinie zu, die Konto B als den Prinzipal identifiziert, der die Rolle übernehmen kann.
  3. Der Administrator von Konto B kann dann Berechtigungen zur Übernahme der Rolle an alle Benutzer in Konto B delegieren. Auf diese Weise können die Benutzer in Konto B Ressourcen in Konto A erstellen oder darauf zugreifen. Der Prinzipal in der Vertrauensrichtlinie kann auch ein AWS Dienstprinzipal sein, wenn Sie einem AWS Dienst Berechtigungen zur Übernahme der Rolle erteilen möchten.

  Weitere Informationen zum Delegieren von Berechtigungen mithilfe von IAM finden Sie unter Zugriffsverwaltung im IAM-Benutzerhandbuch.

Im Folgenden finden Sie eine Beispielrichtlinie, die es dem Benutzer mit der ID ermöglicht, Instanzen für Sie `123456789012` AWS-Konto zu erstellen. Die neue Instance muss eine Optionsgruppe und eine Parametergruppe verwenden, die mit `default` beginnt, und sie muss die Subnetzgruppe `default` verwenden.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCreateDBInstanceOnly",
            "Effect": "Allow",
            "Action": [
```

```
                "rds:CreateDBInstance"
            ],
            "Resource": [
                "arn:aws:rds:*:123456789012:db:test*",
                "arn:aws:rds:*:123456789012:pg:cluster-pg:default*",
                "arn:aws:rds:*:123456789012:subgrp:default"
            ]
        }
    ]
}
```

Weitere Informationen zur Verwendung identitätsbasierter Richtlinien mit Amazon DocumentDB finden Sie unter. Verwendung identitätsbasierter Richtlinien (IAM-Richtlinien) für Amazon DocumentDB Weitere Informationen zu Benutzern, Gruppen, Rollen und Berechtigungen finden Sie im Thema Identitäten (Benutzer, Gruppen und Rollen) im IAM-Benutzerhandbuch.

Resource-based Richtlinien

Andere Dienste, wie Amazon Simple Storage Service (Amazon S3), unterstützen ebenfalls ressourcenbasierte Berechtigungsrichtlinien. Beispielsweise können Sie einem Amazon S3-Bucket eine Richtlinie zuweisen, um die Zugriffsberechtigungen für diesen Bucket zu verwalten. Amazon DocumentDB unterstützt keine ressourcenbasierten Richtlinien.

Spezifizierung von Richtlinienelementen: Aktionen, Auswirkungen, Ressourcen und Prinzipien

Für jede Amazon DocumentDB-Ressource (siheRessourcen und Abläufe von Amazon DocumentDB) definiert der Service eine Reihe von API-Vorgängen. Weitere Informationen finden Sie unter Aktionen. Um Berechtigungen für diese API-Operationen zu gewähren, definiert Amazon DocumentDB eine Reihe von Aktionen, die Sie in einer Richtlinie angeben können. Für das Durchführen einer API-Operation können Berechtigungen für mehrere Aktionen erforderlich sein.

Grundlegende Richtlinienelemente:

- Ressource – In einer Richtlinie wird der Amazon-Ressourcenname (ARN) zur Identifizierung der Ressource verwendet, für die die Richtlinie gilt.
- Aktion – Mit Aktionsschlüsselwörtern geben Sie die Ressourcenoperationen an, die Sie zulassen oder verweigern möchten. Die `rds:DescribeDBInstances`-Berechtigung erteilt dem Benutzer zum Beispiel Berechtigungen zum Ausführen der `DescribeDBInstances`-Operation.

- Auswirkung – Die von Ihnen festgelegte Auswirkung, wenn der Benutzer die jeweilige Aktion anfordert – entweder „allow“ (Zugriffserlaubnis) oder „deny“ (Zugriffsverweigerung). Wenn Sie den Zugriff auf eine Ressource nicht ausdrücklich gestatten ("Allow"), wird er automatisch verweigert. Sie können den Zugriff auf eine Ressource auch explizit verweigern. So können Sie sicherstellen, dass Benutzer nicht darauf zugreifen können, auch wenn der Zugriff durch eine andere Richtlinie gestattet wird.
- Prinzipal – In identitätsbasierten Richtlinien (IAM-Richtlinien) ist der Benutzer, dem die Richtlinie zugewiesen ist, automatisch der Prinzipal. In ressourcenbasierten Richtlinien müssen Sie den Benutzer, das Konto, den Service oder die sonstige Entität angeben, die die Berechtigungen erhalten soll (gilt nur für ressourcenbasierte Richtlinien). Amazon DocumentDB unterstützt keine ressourcenbasierten Richtlinien.

Weitere Informationen zur Syntax und zu Beschreibungen von IAM-Richtlinien finden Sie in der AWS -IAM-Richtlinienreferenz im IAM-Benutzerhandbuch.

Eine Tabelle mit allen Amazon DocumentDB-API-Aktionen und den Ressourcen, für die sie gelten, finden Sie unter. Amazon DocumentDB-API-Berechtigungen: Referenz zu Aktionen, Ressourcen und Bedingungen

Angeben von Bedingungen in einer Richtlinie

Beim Erteilen von Berechtigungen können Sie mithilfe der IAM-Richtliniensyntax die Bedingungen angeben, unter denen die Richtlinie wirksam werden soll. Beispielsweise kann festgelegt werden, dass eine Richtlinie erst ab einem bestimmten Datum gilt. Weitere Informationen zum Angeben von Bedingungen in einer Richtliniensyntax finden Sie im Thema Bedingung im IAM Benutzerhandbuch.

Bedingungen werden mithilfe vordefinierter Bedingungsschlüssel formuliert. Amazon DocumentDB hat keine dienstspezifischen Kontextschlüssel, die in einer IAM-Richtlinie verwendet werden können. Eine Liste der globalen Bedingungskontextschlüssel, die für alle Dienste verfügbar sind, finden Sie im IAM-Benutzerhandbuch unter Verfügbare globale Bedingungsschlüssel.

Verwendung identitätsbasierter Richtlinien (IAM-Richtlinien) für Amazon DocumentDB

> **Important**
>
> Für bestimmte Verwaltungsfunktionen verwendet Amazon DocumentDB Betriebstechnologie, die mit Amazon RDS gemeinsam genutzt wird. Amazon DocumentDB-Konsole und API-Aufrufe werden als Aufrufe an die Amazon RDS-API protokolliert. AWS CLI
> Lesen Sie zunächst die einführenden Themen, in denen die grundlegenden Konzepte und Optionen für die Verwaltung des Zugriffs auf Ihre Amazon DocumentDB-Ressourcen erläutert werden. Weitere Informationen finden Sie unter Verwaltung der Zugriffsberechtigungen für Ihre Amazon DocumentDB-Ressourcen.

In diesem Thema finden Sie Beispiele für identitätsbasierte Richtlinien, in denen ein Kontoadministrator den IAM-Identitäten (Benutzer, Gruppen und Rollen) Berechtigungsrichtlinien anfügen kann.

Das Folgende ist ein Beispiel für eine IAM-Richtlinie.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCreateDBInstanceOnly",
            "Effect": "Allow",
            "Action": [
                "rds:CreateDBInstance"
            ],
            "Resource": [
                "arn:aws:rds:*:123456789012:db:test*",
                "arn:aws:rds:*:123456789012:pg:cluster-pg:default*",
                "arn:aws:rds:*:123456789012:subgrp:default"
            ]
        }
    ]
}
```

Die Richtlinie ist ein einzelnes Statement, das die folgenden Berechtigungen für den IAM-Benutzer bestimmt:

- Die Richtlinie ermöglicht es dem IAM-Benutzer, mithilfe der Aktion CreateDBInstance eine Instanz zu erstellen (dies gilt auch für den Vorgang https://docs.aws.amazon.com/cli/latest/reference/rds/create-db-instance.html AWS CLI create-db-instance und den). AWS-Managementkonsole
- Das Element `Resource` gibt an, dass der Benutzer auf oder mit Ressourcen Aktionen ausführen kann. Sie geben Ressourcen über einen Amazon-Ressourcennamen (ARN) an. Dieser ARN beinhaltet den Namen des Dienstes, zu dem die Ressource gehört (`rds`), den AWS-Region (*steht in diesem Beispiel für eine beliebige Region), die Benutzerkontonummer (123456789012ist in diesem Beispiel die Benutzer-ID) und den Typ der Ressource.

  Das `Resource`-Element im Beispiel gibt für den Benutzer die folgenden richtlinienbezogenen Einschränkungen für die Ressourcen an:

  - Die Instance-Kennung für die neue Instance muss mit `test` beginnen (zum Beispiel `testCustomerData1`, `test-region2-data`).
  - Die Parametergruppe für die neue Instance muss mit `default` beginnen.
  - Die Subnetzgruppe für die neue Instance muss mit `default` beginnen.

Das Element `Principal` ist in der Richtlinie nicht angegeben, da in identitätsbasierten Richtlinien die Angabe des Prinzipals als Empfänger der Berechtigung nicht erforderlich ist. Wenn Sie einem Benutzer eine Richtlinie zuweisen, ist der Benutzer automatisch der Prinzipal. Wird die Berechtigungsrichtlinie einer IAM-Rolle angefügt, erhält der in der Vertrauensrichtlinie der Rolle angegebene Prinzipal die Berechtigungen.

Eine Tabelle mit allen Amazon DocumentDB-API-Vorgängen und den Ressourcen, für die sie gelten, finden Sie unterAmazon DocumentDB-API-Berechtigungen: Referenz zu Aktionen, Ressourcen und Bedingungen.

Für die Verwendung der Amazon DocumentDB-Konsole sind Berechtigungen erforderlich

Damit ein Benutzer mit der Amazon DocumentDB-Konsole arbeiten kann, muss dieser Benutzer über ein Mindestmaß an Berechtigungen verfügen. Diese Berechtigungen ermöglichen es dem Benutzer,

die Amazon DocumentDB-Ressourcen für ihn zu beschreiben AWS-Konto und andere zugehörige Informationen bereitzustellen, einschließlich Amazon EC2-Sicherheits- und Netzwerkinformationen.

Wenn Sie eine IAM-Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Benutzer mit dieser IAM-Richtlinie. Um sicherzustellen, dass diese Benutzer die Amazon DocumentDB-Konsole weiterhin verwenden können, fügen Sie dem Benutzer auch die `AmazonDocDBConsoleFullAccess` verwaltete Richtlinie hinzu, wie unter beschrieben. AWS verwaltete Richtlinien für Amazon DocumentDB

Sie müssen Benutzern, die nur die Amazon DocumentDB-API aufrufen, keine Mindestberechtigungen für die Konsole gewähren. AWS CLI

Beispiele für vom Kunden verwaltete Richtlinien

In diesem Abschnitt finden Sie Beispiele für Benutzerrichtlinien, die Berechtigungen für verschiedene Amazon DocumentDB-Aktionen gewähren. Diese Richtlinien funktionieren, wenn Sie Amazon DocumentDB-API-Aktionen, AWS SDKs oder die verwenden. AWS CLI Bei Verwendung der Konsole müssen Sie zusätzliche konsolenspezifische Berechtigungen erteilen, die im Abschnitt Für die Verwendung der Amazon DocumentDB-Konsole sind Berechtigungen erforderlich erläutert werden.

Für bestimmte Verwaltungsfunktionen verwendet Amazon DocumentDB Betriebstechnologie, die mit Amazon Relational Database Service (Amazon RDS) und Amazon Neptune gemeinsam genutzt wird.

> **Note**
>
> Alle Beispiele verwenden die Region USA Ost (Nord-Virginia) (`us-east-1`) und enthalten fiktive Konto-IDs.

Beispiele

- Beispiel 1: Erlauben Sie einem Benutzer, eine beliebige Beschreibungsaktion für eine beliebige Amazon DocumentDB-Ressource auszuführen
- Beispiel 2: Verhindern Sie, dass ein Benutzer eine Instance löscht
- Beispiel 3: Verhindern Sie, dass ein Benutzer einen Cluster erstellt, sofern die Speicherverschlüsselung nicht aktiviert ist

Beispiel 1: Erlauben Sie einem Benutzer, eine beliebige Beschreibungsaktion für eine beliebige Amazon DocumentDB-Ressource auszuführen

Die folgende Berechtigungsrichtlinie gewährt Berechtigungen für einen Benutzer, alle Aktionen auszuführen, die mit beginne `Describe`. Diese Aktionen zeigen Informationen über eine Amazon DocumentDB-Ressource, z. B. eine Instance. Das Platzhalterzeichen (*) im `Resource` Element gibt an, dass die Aktionen für alle Amazon DocumentDB-Ressourcen zulässig sind, die dem Konto gehören.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowRDSDescribe",
         "Effect":"Allow",
         "Action":"rds:Describe*",
         "Resource":"*"
      }
   ]
}
```

Beispiel 2: Verhindern Sie, dass ein Benutzer eine Instance löscht

Die folgenden Berechtigungsrichtlinien erteilen Berechtigungen, um einen Benutzer davon abzuhalten, eine bestimmte Instance zu löschen. Beispielsweise möchten Sie jedem Benutzer, der kein Administrator ist, verbieten, Ihre Produktions-Instances zu löschen.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"DenyDelete1",
         "Effect":"Deny",
         "Action":"rds:DeleteDBInstance",
         "Resource":"arn:aws:rds:us-east-1:123456789012:db:my-db-instance"
```

```
      }
   ]
}
```

Beispiel 3: Verhindern Sie, dass ein Benutzer einen Cluster erstellt, sofern die Speicherverschlüsselung nicht aktiviert ist

Die folgende Berechtigungsrichtlinie verweigert einem Benutzer die Erlaubnis, einen Amazon DocumentDB-Cluster zu erstellen, sofern die Speicherverschlüsselung nicht aktiviert ist.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "PreventUnencryptedDocumentDB",
         "Effect": "Deny",
         "Action": "RDS:CreateDBCluster",
         "Condition": {
         "Bool": {
         "rds:StorageEncrypted": "false"
      },
         "StringEquals": {
         "rds:DatabaseEngine": "docdb"
         }
      },
      "Resource": "*"
      }
   ]
}
```

AWS verwaltete Richtlinien für Amazon DocumentDB

Um Benutzern, Gruppen und Rollen Berechtigungen hinzuzufügen, ist es einfacher, AWS verwaltete Richtlinien zu verwenden, als Richtlinien selbst zu schreiben. Es erfordert Zeit und Fachwissen, um von Kunden verwaltete IAM-Richtlinien zu erstellen, die Ihrem Team nur die benötigten Berechtigungen bieten. Um schnell loszulegen, können Sie unsere AWS verwalteten Richtlinien verwenden. Diese Richtlinien decken allgemeine Anwendungsfälle ab und sind in Ihrem AWS-Konto

verfügbar. Weitere Informationen zu AWS verwalteten Richtlinien finden Sie unter AWS Verwaltete Richtlinien im IAM-Benutzerhandbuch.

AWS Dienste verwalten und aktualisieren AWS verwaltete Richtlinien. Sie können die Berechtigungen in AWS verwalteten Richtlinien nicht ändern. Dienste fügen einer AWS verwalteten Richtlinie gelegentlich zusätzliche Berechtigungen hinzu, um neue Funktionen zu unterstützen. Diese Art von Update betrifft alle Identitäten (Benutzer, Gruppen und Rollen), an welche die Richtlinie angehängt ist. Dienste aktualisieren eine AWS verwaltete Richtlinie am ehesten, wenn eine neue Funktion gestartet wird oder wenn neue Funktionen verfügbar werden. Dienste entfernen keine Berechtigungen aus einer AWS verwalteten Richtlinie, sodass durch Richtlinienaktualisierungen Ihre bestehenden Berechtigungen nicht beeinträchtigt werden.

AWS Unterstützt außerdem verwaltete Richtlinien für Jobfunktionen, die sich über mehrere Dienste erstrecken. Die `ViewOnlyAccess` AWS verwaltete Richtlinie bietet beispielsweise schreibgeschützten Zugriff auf viele AWS Dienste und Ressourcen. Wenn ein Dienst eine neue Funktion startet, werden Leseberechtigungen für neue Operationen und Ressourcen AWS hinzugefügt. Eine Liste und Beschreibungen der Richtlinien für Auftragsfunktionen finden Sie in Verwaltete AWS -Richtlinien für Auftragsfunktionen im IAM-Leitfaden.

Die folgenden AWS verwalteten Richtlinien, die Sie Benutzern in Ihrem Konto zuordnen können, sind spezifisch für Amazon DocumentDB:

- AmazonDocDBFullAccess— Gewährt vollen Zugriff auf alle Amazon DocumentDB-Ressourcen für das Stammverzeichnis. AWS-Konto
- AmazonDocDBReadOnlyAccess— Gewährt schreibgeschützten Zugriff auf alle Amazon DocumentDB-Ressourcen für das Stammverzeichnis. AWS-Konto
- AmazonDocDBConsoleFullAccess— Gewährt vollen Zugriff auf die Verwaltung der elastischen Cluster-Ressourcen von Amazon DocumentDB und Amazon DocumentDB mithilfe von. AWS-Managementkonsole
- AmazonDocDBElasticReadOnlyAccess— Gewährt schreibgeschützten Zugriff auf alle Amazon DocumentDB Elastic Cluster-Ressourcen für das Stammverzeichnis. AWS-Konto
- AmazonDocDBElasticFullAccess— Gewährt vollen Zugriff auf alle Amazon DocumentDB Elastic Cluster-Ressourcen für das Stammverzeichnis. AWS-Konto

AmazonDocDBFullAccess

Diese Richtlinie gewährt Administratorberechtigungen, die einem Prinzipal vollen Zugriff auf alle Amazon DocumentDB-Aktionen ermöglichen. Die Berechtigungen in dieser Richtlinie sind wie folgt gruppiert:

- Die Amazon DocumentDB-Berechtigungen ermöglichen alle Amazon DocumentDB-Aktionen.
- Einige der Amazon-EC2-Berechtigungen in dieser Richtlinie sind erforderlich, um die übergebenen Ressourcen in einer API-Anfrage zu validieren. Dadurch wird sichergestellt, dass Amazon DocumentDB die Ressourcen erfolgreich mit einem Cluster verwenden kann. Die übrigen Amazon EC2-Berechtigungen in dieser Richtlinie ermöglichen es Amazon DocumentDB, AWS Ressourcen zu erstellen, die benötigt werden, damit Sie eine Verbindung zu Ihren Clustern herstellen können.
- Die Amazon DocumentDB-Berechtigungen werden bei API-Aufrufen verwendet, um die in einer Anfrage übergebenen Ressourcen zu validieren. Sie sind erforderlich, damit Amazon DocumentDB den übergebenen Schlüssel mit dem Amazon DocumentDB-Cluster verwenden kann.
- Die CloudWatch Protokolle sind erforderlich, damit Amazon DocumentDB sicherstellen kann, dass die Ziele der Protokollzustellung erreichbar sind und dass sie für die Verwendung von Broker-Protokollen gültig sind.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "rds:AddRoleToDBCluster",
                "rds:AddSourceIdentifierToSubscription",
                "rds:AddTagsToResource",
                "rds:ApplyPendingMaintenanceAction",
                "rds:CopyDBClusterParameterGroup",
                "rds:CopyDBClusterSnapshot",
                "rds:CopyDBParameterGroup",
                "rds:CreateDBCluster",
                "rds:CreateDBClusterParameterGroup",
                "rds:CreateDBClusterSnapshot",
                "rds:CreateDBInstance",
                "rds:CreateDBParameterGroup",
                "rds:CreateDBSubnetGroup",
```

```
                "rds:CreateEventSubscription",
                "rds:DeleteDBCluster",
                "rds:DeleteDBClusterParameterGroup",
                "rds:DeleteDBClusterSnapshot",
                "rds:DeleteDBInstance",
                "rds:DeleteDBParameterGroup",
                "rds:DeleteDBSubnetGroup",
                "rds:DeleteEventSubscription",
                "rds:DescribeAccountAttributes",
                "rds:DescribeCertificates",
                "rds:DescribeDBClusterParameterGroups",
                "rds:DescribeDBClusterParameters",
                "rds:DescribeDBClusterSnapshotAttributes",
                "rds:DescribeDBClusterSnapshots",
                "rds:DescribeDBClusters",
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeDBLogFiles",
                "rds:DescribeDBParameterGroups",
                "rds:DescribeDBParameters",
                "rds:DescribeDBSecurityGroups",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribeEngineDefaultClusterParameters",
                "rds:DescribeEngineDefaultParameters",
                "rds:DescribeEventCategories",
                "rds:DescribeEventSubscriptions",
                "rds:DescribeEvents",
                "rds:DescribeOptionGroups",
                "rds:DescribeOrderableDBInstanceOptions",
                "rds:DescribePendingMaintenanceActions",
                "rds:DescribeValidDBInstanceModifications",
                "rds:DownloadDBLogFilePortion",
                "rds:FailoverDBCluster",
                "rds:ListTagsForResource",
                "rds:ModifyDBCluster",
                "rds:ModifyDBClusterParameterGroup",
                "rds:ModifyDBClusterSnapshotAttribute",
                "rds:ModifyDBInstance",
                "rds:ModifyDBParameterGroup",
                "rds:ModifyDBSubnetGroup",
                "rds:ModifyEventSubscription",
                "rds:PromoteReadReplicaDBCluster",
                "rds:RebootDBInstance",
                "rds:RemoveRoleFromDBCluster",
```

```
                "rds:RemoveSourceIdentifierFromSubscription",
                "rds:RemoveTagsFromResource",
                "rds:ResetDBClusterParameterGroup",
                "rds:ResetDBParameterGroup",
                "rds:RestoreDBClusterFromSnapshot",
                "rds:RestoreDBClusterToPointInTime"
            ],
            "Effect": "Allow",
            "Resource": [
                "*"
            ]
        },
        {
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcs",
                "kms:ListAliases",
                "kms:ListKeyPolicies",
                "kms:ListKeys",
                "kms:ListRetirableGrants",
                "logs:DescribeLogStreams",
                "logs:GetLogEvents",
                "sns:ListSubscriptions",
                "sns:ListTopics",
                "sns:Publish"
            ],
            "Effect": "Allow",
            "Resource": [
                "*"
            ]
        },
        {
            "Action": "iam:CreateServiceLinkedRole",
            "Effect": "Allow",
            "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
            "Condition": {
                "StringLike": {
```

```
                    "iam:AWS ServiceName": "rds.amazonaws.com"
                }
            }
        }
    ]
}
```

AmazonDocDBReadOnlyAccess

Diese Richtlinie gewährt nur Leseberechtigungen, die es Benutzern ermöglichen, Informationen in Amazon DocumentDB einzusehen. Prinzipale, an die diese Richtlinie angehängt ist, können keine Aktualisierungen vornehmen oder vorhandene Ressourcen löschen. Sie können auch keine neuen Amazon DocumentDB-Ressourcen erstellen. Prinzipale mit diesen Berechtigungen können beispielsweise die Liste der Cluster und Konfigurationen, die mit ihrem Konto verknüpft sind, einsehen, aber nicht die Konfiguration oder Einstellungen von Clustern ändern. Die Berechtigungen in dieser Richtlinie sind wie folgt gruppiert:

- Amazon DocumentDB-Berechtigungen ermöglichen es Ihnen, Amazon DocumentDB-Ressourcen aufzulisten, sie zu beschreiben und Informationen über sie abzurufen.
- Amazon EC2-Berechtigungen werden verwendet, um die Amazon VPC, Subnetze, Sicherheitsgruppen und ENIs zu beschreiben, die einem Cluster zugeordnet sind.
- Eine Amazon DocumentDB-Berechtigung wird verwendet, um den Schlüssel zu beschreiben, der dem Cluster zugeordnet ist.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "rds:DescribeAccountAttributes",
                "rds:DescribeCertificates",
                "rds:DescribeDBClusterParameterGroups",
                "rds:DescribeDBClusterParameters",
                "rds:DescribeDBClusterSnapshotAttributes",
                "rds:DescribeDBClusterSnapshots",
                "rds:DescribeDBClusters",
```

```
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeDBLogFiles",
                "rds:DescribeDBParameterGroups",
                "rds:DescribeDBParameters",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribeEventCategories",
                "rds:DescribeEventSubscriptions",
                "rds:DescribeEvents",
                "rds:DescribeOrderableDBInstanceOptions",
                "rds:DescribePendingMaintenanceActions",
                "rds:DownloadDBLogFilePortion",
                "rds:ListTagsForResource"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeInternetGateways",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcs"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "kms:ListKeys",
                "kms:ListRetirableGrants",
                "kms:ListAliases",
                "kms:ListKeyPolicies"
```

```
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "logs:DescribeLogStreams",
                "logs:GetLogEvents"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:logs:*:*:log-group:/aws/rds/*:log-stream:*",
                "arn:aws:logs:*:*:log-group:/aws/docdb/*:log-stream:*"
            ]
        }
    ]
}
```

AmazonDocDBConsoleFullAccess

Gewährt vollen Zugriff auf die Verwaltung von Amazon DocumentDB-Ressourcen mithilfe der folgenden Formulare AWS-Managementkonsole :

- Die Amazon DocumentDB-Berechtigungen, um alle Amazon DocumentDB- und Amazon DocumentDB-Cluster-Aktionen zuzulassen.
- Einige der Amazon-EC2-Berechtigungen in dieser Richtlinie sind erforderlich, um die übergebenen Ressourcen in einer API-Anfrage zu validieren. Dadurch wird sichergestellt, dass Amazon DocumentDB die Ressourcen für die Bereitstellung und Wartung des Clusters erfolgreich nutzen kann. Die übrigen Amazon EC2-Berechtigungen in dieser Richtlinie ermöglichen es Amazon DocumentDB, AWS Ressourcen zu erstellen, die erforderlich sind, um Ihnen die Verbindung zu Ihren Clustern wie vpcEndpoint zu ermöglichen.
- AWS KMS Berechtigungen werden bei API-Aufrufen verwendet, um die in einer AWS KMS Anfrage übergebenen Ressourcen zu validieren. Sie sind erforderlich, damit Amazon DocumentDB den übergebenen Schlüssel verwenden kann, um die ruhenden Daten mit dem Amazon DocumentDB Elastic Cluster zu verschlüsseln und zu entschlüsseln.
- Die CloudWatch Protokolle sind erforderlich, damit Amazon DocumentDB sicherstellen kann, dass die Ziele der Protokollzustellung erreichbar sind und dass sie für die Verwendung von Protokollen zur Prüfung und Profilerstellung gültig sind.

- Secrets Manager-Berechtigungen sind erforderlich, um ein bestimmtes Secret zu validieren und es zu verwenden, um den Admin-Benutzer für Amazon DocumentDB Elastic Clusters einzurichten.
- Amazon RDS-Berechtigungen sind für Amazon DocumentDB-Clusterverwaltungsaktionen erforderlich. Für bestimmte Verwaltungsfunktionen verwendet Amazon DocumentDB Betriebstechnologie, die mit Amazon RDS gemeinsam genutzt wird.
- SNS-Berechtigungen ermöglichen es Principals, Amazon Simple Notification Service (Amazon SNS) -Abonnements und -Themen zu nutzen und Amazon SNS-Nachrichten zu veröffentlichen.
- Für die Erstellung der serviceverknüpften Rollen, die für die Veröffentlichung von Metriken und Protokollen erforderlich sind, sind IAM-Berechtigungen erforderlich.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DocdbSids",
            "Effect": "Allow",
            "Action": [
                "docdb-elastic:CreateCluster",
                "docdb-elastic:UpdateCluster",
                "docdb-elastic:GetCluster",
                "docdb-elastic:DeleteCluster",
                "docdb-elastic:ListClusters",
                "docdb-elastic:CreateClusterSnapshot",
                "docdb-elastic:GetClusterSnapshot",
                "docdb-elastic:DeleteClusterSnapshot",
                "docdb-elastic:ListClusterSnapshots",
                "docdb-elastic:RestoreClusterFromSnapshot",
                "docdb-elastic:TagResource",
                "docdb-elastic:UntagResource",
                "docdb-elastic:ListTagsForResource",
                "docdb-elastic:CopyClusterSnapshot",
                "docdb-elastic:StartCluster",
                "docdb-elastic:StopCluster",
                "docdb-elastic:GetPendingMaintenanceAction",
                "docdb-elastic:ListPendingMaintenanceActions",
                "docdb-elastic:ApplyPendingMaintenanceAction",
                "rds:AddRoleToDBCluster",
                "rds:AddSourceIdentifierToSubscription",
```

```
                "rds:AddTagsToResource",
                "rds:ApplyPendingMaintenanceAction",
                "rds:CopyDBClusterParameterGroup",
                "rds:CopyDBClusterSnapshot",
                "rds:CopyDBParameterGroup",
                "rds:CreateDBCluster",
                "rds:CreateDBClusterParameterGroup",
                "rds:CreateDBClusterSnapshot",
                "rds:CreateDBInstance",
                "rds:CreateDBParameterGroup",
                "rds:CreateDBSubnetGroup",
                "rds:CreateEventSubscription",
                "rds:CreateGlobalCluster",
                "rds:DeleteDBCluster",
                "rds:DeleteDBClusterParameterGroup",
                "rds:DeleteDBClusterSnapshot",
                "rds:DeleteDBInstance",
                "rds:DeleteDBParameterGroup",
                "rds:DeleteDBSubnetGroup",
                "rds:DeleteEventSubscription",
                "rds:DeleteGlobalCluster",
                "rds:DescribeAccountAttributes",
                "rds:DescribeCertificates",
                "rds:DescribeDBClusterParameterGroups",
                "rds:DescribeDBClusterParameters",
                "rds:DescribeDBClusterSnapshotAttributes",
                "rds:DescribeDBClusterSnapshots",
                "rds:DescribeDBClusters",
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeDBLogFiles",
                "rds:DescribeDBParameterGroups",
                "rds:DescribeDBParameters",
                "rds:DescribeDBSecurityGroups",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribeEngineDefaultClusterParameters",
                "rds:DescribeEngineDefaultParameters",
                "rds:DescribeEventCategories",
                "rds:DescribeEventSubscriptions",
                "rds:DescribeEvents",
                "rds:DescribeGlobalClusters",
                "rds:DescribeOptionGroups",
                "rds:DescribeOrderableDBInstanceOptions",
                "rds:DescribePendingMaintenanceActions",
```

```
                "rds:DescribeValidDBInstanceModifications",
                "rds:DownloadDBLogFilePortion",
                "rds:FailoverDBCluster",
                "rds:ListTagsForResource",
                "rds:ModifyDBCluster",
                "rds:ModifyDBClusterParameterGroup",
                "rds:ModifyDBClusterSnapshotAttribute",
                "rds:ModifyDBInstance",
                "rds:ModifyDBParameterGroup",
                "rds:ModifyDBSubnetGroup",
                "rds:ModifyEventSubscription",
                "rds:ModifyGlobalCluster",
                "rds:PromoteReadReplicaDBCluster",
                "rds:RebootDBInstance",
                "rds:RemoveFromGlobalCluster",
                "rds:RemoveRoleFromDBCluster",
                "rds:RemoveSourceIdentifierFromSubscription",
                "rds:RemoveTagsFromResource",
                "rds:ResetDBClusterParameterGroup",
                "rds:ResetDBParameterGroup",
                "rds:RestoreDBClusterFromSnapshot",
                "rds:RestoreDBClusterToPointInTime"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "DependencySids",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "cloudwatch:GetMetricData",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics",
                "ec2:AllocateAddress",
                "ec2:AssignIpv6Addresses",
                "ec2:AssignPrivateIpAddresses",
                "ec2:AssociateAddress",
                "ec2:AssociateRouteTable",
                "ec2:AssociateSubnetCidrBlock",
                "ec2:AssociateVpcCidrBlock",
                "ec2:AttachInternetGateway",
                "ec2:AttachNetworkInterface",
```

```
                "ec2:CreateCustomerGateway",
                "ec2:CreateDefaultSubnet",
                "ec2:CreateDefaultVpc",
                "ec2:CreateInternetGateway",
                "ec2:CreateNatGateway",
                "ec2:CreateNetworkInterface",
                "ec2:CreateRoute",
                "ec2:CreateRouteTable",
                "ec2:CreateSecurityGroup",
                "ec2:CreateSubnet",
                "ec2:CreateVpc",
                "ec2:CreateVpcEndpoint",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAddresses",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeCustomerGateways",
                "ec2:DescribeInstances",
                "ec2:DescribeNatGateways",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribePrefixLists",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroupReferences",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeVpcs",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:ModifySubnetAttribute",
                "ec2:ModifyVpcAttribute",
                "ec2:ModifyVpcEndpoint",
                "kms:DescribeKey",
                "kms:ListAliases",
                "kms:ListKeyPolicies",
                "kms:ListKeys",
                "kms:ListRetirableGrants",
                "logs:DescribeLogStreams",
                "logs:GetLogEvents",
                "sns:ListSubscriptions",
                "sns:ListTopics",
                "sns:Publish"
            ],
            "Resource": [
                "*"
```

```
            ]
        },
        {
            "Sid": "DocdbSLRSid",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "rds.amazonaws.com"
                }
            }
        },
        {
            "Sid": "DocdbElasticSLRSid",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/docdb-
elastic.amazonaws.com/AWSServiceRoleForDocDB-Elastic",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "docdb-elastic.amazonaws.com"
                }
            }
        }
    ]
}
```

AmazonDocDBElasticReadOnlyAccess

Diese Richtlinie gewährt nur Leseberechtigungen, die es Benutzern ermöglichen, Elastic Cluster-Informationen in Amazon DocumentDB anzuzeigen. Principals, an die diese Richtlinie angehängt ist, können keine Aktualisierungen vornehmen oder vorhandene Ressourcen löschen, und sie können auch keine neuen Amazon DocumentDB-Ressourcen erstellen. Prinzipale mit diesen Berechtigungen können beispielsweise die Liste der Cluster und Konfigurationen, die mit ihrem Konto verknüpft sind, einsehen, aber nicht die Konfiguration oder Einstellungen von Clustern ändern. Die Berechtigungen in dieser Richtlinie sind wie folgt gruppiert:

- Amazon DocumentDB Elastic Cluster-Berechtigungen ermöglichen es Ihnen, Amazon DocumentDB Elastic Cluster-Ressourcen aufzulisten, sie zu beschreiben und Informationen über sie zu erhalten.
- CloudWatch Berechtigungen werden verwendet, um Servicemetriken zu überprüfen.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "docdb-elastic:ListClusters",
            "docdb-elastic:GetCluster",
            "docdb-elastic:ListClusterSnapshots",
            "docdb-elastic:GetClusterSnapshot",
            "docdb-elastic:ListTagsForResource"
         ],
         "Resource": "*"
      },
      {
         "Effect": "Allow",
         "Action": [
            "cloudwatch:GetMetricData",
            "cloudwatch:ListMetrics",
            "cloudwatch:GetMetricStatistics"
         ],
         "Resource": "*"
      }
   ]
}
```

AmazonDocDBElasticFullAccess

Diese Richtlinie gewährt Administratorberechtigungen, die einem Prinzipal vollen Zugriff auf alle Amazon DocumentDB-Aktionen für Amazon DocumentDB Elastic Cluster ermöglichen.

Diese Richtlinie verwendet AWS Tags (https://docs.aws.amazon.com/tag-editor/latest/userguide/tagging.html) innerhalb von Bedingungen, um den Zugriff auf Ressourcen einzuschränken. Wenn

Sie ein Geheimnis verwenden, muss es mit einem Tag-Schlüssel `DocDBElasticFullAccess` und einem Tag-Wert versehen werden. Wenn Sie einen vom Kunden verwalteten Schlüssel verwenden, muss dieser mit einem Tag-Schlüssel `DocDBElasticFullAccess` und einem Tag-Wert versehen werden.

Die Berechtigungen in dieser Richtlinie sind wie folgt gruppiert:

- Amazon DocumentDB Elastic Cluster-Berechtigungen ermöglichen alle Amazon DocumentDB-Aktionen.
- Einige der Amazon-EC2-Berechtigungen in dieser Richtlinie sind erforderlich, um die übergebenen Ressourcen in einer API-Anfrage zu validieren. Dadurch wird sichergestellt, dass Amazon DocumentDB die Ressourcen für die Bereitstellung und Wartung des Clusters erfolgreich nutzen kann. Die übrigen Amazon EC2-Berechtigungen in dieser Richtlinie ermöglichen es Amazon DocumentDB, AWS Ressourcen zu erstellen, die erforderlich sind, damit Sie wie bei einem VPC-Endpunkt eine Verbindung zu Ihren Clustern herstellen können.
- AWS KMS Berechtigungen sind erforderlich, damit Amazon DocumentDB den übergebenen Schlüssel verwenden kann, um die ruhenden Daten innerhalb des elastischen Amazon DocumentDB-Clusters zu verschlüsseln und zu entschlüsseln.

  > **Note**
  >
  > Der vom Kunden verwaltete Schlüssel muss ein Tag mit einem Schlüssel `DocDBElasticFullAccess` und einem Tag-Wert haben.

- SecretsManager Berechtigungen sind erforderlich, um einen bestimmten Schlüssel zu validieren und ihn zu verwenden, um den Admin-Benutzer für Amazon DocumentDB Elastic Clusters einzurichten.

  > **Note**
  >
  > Das verwendete Geheimnis muss ein Tag mit einem Schlüssel `DocDBElasticFullAccess` und einem Tag-Wert haben.

- Für die Erstellung der serviceverknüpften Rollen, die für die Veröffentlichung von Metriken und Protokollen erforderlich sind, sind IAM-Berechtigungen erforderlich.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DocdbElasticSid",
            "Effect": "Allow",
            "Action": [
                "docdb-elastic:CreateCluster",
                "docdb-elastic:UpdateCluster",
                "docdb-elastic:GetCluster",
                "docdb-elastic:DeleteCluster",
                "docdb-elastic:ListClusters",
                "docdb-elastic:CreateClusterSnapshot",
                "docdb-elastic:GetClusterSnapshot",
                "docdb-elastic:DeleteClusterSnapshot",
                "docdb-elastic:ListClusterSnapshots",
                "docdb-elastic:RestoreClusterFromSnapshot",
                "docdb-elastic:TagResource",
                "docdb-elastic:UntagResource",
                "docdb-elastic:ListTagsForResource",
                "docdb-elastic:CopyClusterSnapshot",
                "docdb-elastic:StartCluster",
                "docdb-elastic:StopCluster",
                "docdb-elastic:GetPendingMaintenanceAction",
                "docdb-elastic:ListPendingMaintenanceActions",
                "docdb-elastic:ApplyPendingMaintenanceAction"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "EC2Sid",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpoint",
                "ec2:DescribeVpcEndpoints",
                "ec2:DeleteVpcEndpoints",
                "ec2:ModifyVpcEndpoint",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeSecurityGroups",
```

```
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DescribeAvailabilityZones",
                "secretsmanager:ListSecrets"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "docdb-elastic.amazonaws.com"
                }
            }
        },
        {
            "Sid": "KMSSid",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": [
                        "docdb-elastic.*.amazonaws.com"
                    ],
                    "aws:ResourceTag/DocDBElasticFullAccess": "*"
                }
            }
        },
        {
            "Sid": "KMSGrantSid",
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/DocDBElasticFullAccess": "*",
                    "kms:ViaService": [
                        "docdb-elastic.*.amazonaws.com"
```

```
                    ]
                },
                "Bool": {
                    "kms:GrantIsForAWSResource": true
                }
            }
        },
        {
            "Sid": "SecretManagerSid",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:ListSecretVersionIds",
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:GetResourcePolicy"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "secretsmanager:ResourceTag/DocDBElasticFullAccess": "*"
                },
                "StringEquals": {
                    "aws:CalledViaFirst": "docdb-elastic.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CloudwatchSid",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:ListMetrics",
                "cloudwatch:GetMetricStatistics"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "SLRSid",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/docdb-
elastic.amazonaws.com/AWSServiceRoleForDocDB-Elastic",
```

```
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "docdb-elastic.amazonaws.com"
                }
            }
        }
    ]
}
```

AmazonDocDB-ElasticServiceRolePolicy

Sie können keine Elemente `AmazonDocDBElasticServiceRolePolicy` an Ihre AWS Identity and Access Management Entitäten anhängen. Diese Richtlinie ist an eine serviceverknüpfte Rolle angehängt, die es Amazon DocumentDB ermöglicht, Aktionen in Ihrem Namen durchzuführen. Weitere Informationen finden Sie unter Service-linked Rollen in elastischen Clustern.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": [
                        "AWS/DocDB-Elastic"
                    ]
                }
            }
        }
    ]
}
```

Amazon DocumentDB aktualisiert auf AWS Verwaltete Richtlinien

Änderungen	Beschreibung	Date
AmazonDocDBElasticFullAccess, AmazonDocDBConsoleFullAccess - Ändern	Die Richtlinien wurden aktualisiert, um ausstehende Wartungsmaßnahmen hinzuzufügen.	2/11/2025
AmazonDocDBElasticFullAccess, AmazonDocDBConsoleFullAccess - Veränderung	Die Richtlinien wurden aktualisiert, um start/stop Cluster-Snapshot-Aktionen hinzuzufügen und Cluster-Snapshots zu kopieren.	2/21/2024
AmazonDocDBElasticReadOnlyAccess, AmazonDocDBElasticFullAccess - Veränderung	Richtlinien wurden aktualisiert, um `cloudwatch:GetMetricData` Aktionen hinzuzufügen.	6/21/2023
AmazonDocDBElasticReadOnlyAccess – Neue Richtlinie	Neue verwaltete Richtlinie für elastische Amazon DocumentDB-Cluster.	6/8/2023
AmazonDocDBElasticFullAccess – Neue Richtlinie	Neue verwaltete Richtlinie für elastische Amazon DocumentDB-Cluster.	6/5/2023
AmazonDocDB-ElasticServiceRolePolicy – Neue Richtlinie	Amazon DocumentDB erstellt eine neue AWS ServiceRoleForDocDB-Elastic serviceverknüpfte Rolle für Amazon DocumentDB Elastic Clusters.	11/30/2022
AmazonDocDBConsoleFullAccess- Veränderung	Die Richtlinie wurde aktualisiert, um globale und elastische Cluster-Berechtigungen	11/30/2022

Änderungen	Beschreibung	Date
	für Amazon DocumentDB hinzuzufügen.	
AmazonDocDBConsoleFullAccess,AmazonDocDBFullAccess, AmazonDocDBReadOnlyAccess - Neue Richtlinie	Start des Dienstes.	1/19/2017

Amazon DocumentDB-API-Berechtigungen: Referenz zu Aktionen, Ressourcen und Bedingungen

Verwenden Sie die folgenden Abschnitte als Referenz, wenn Sie Berechtigungsrichtlinien einrichten Verwendung identitätsbasierter Richtlinien (IAM-Richtlinien) für Amazon DocumentDB und schreiben, die Sie an eine IAM-Identität anhängen können (identitätsbasierte Richtlinien).

Im Folgenden sind alle Amazon DocumentDB-API-Operationen aufgeführt. In der Liste sind die entsprechenden Aktionen enthalten, für die Sie Berechtigungen zur Ausführung der Aktion gewähren können, die AWS Ressource, für die Sie die Berechtigungen erteilen können, und Bedingungsschlüssel, die Sie für eine detaillierte Zugriffskontrolle einbeziehen können. Sie geben die Aktionen im Feld `Action` der Richtlinie, den Ressourcenwert im Feld `Resource` der Richtlinie und die Bedingungen im Feld `Condition` der Richtlinie an. Weitere Informationen über Bedingungen finden Sie unter Angeben von Bedingungen in einer Richtlinie.

Sie können in Ihren Amazon DocumentDB-Richtlinien allgemeine Bedingungsschlüssel verwenden AWS, um Bedingungen auszudrücken. Eine vollständige Liste der AWS-wide Keys finden Sie im IAM-Benutzerhandbuch unter Verfügbare Schlüssel.

Sie können IAM-Richtlinien mit dem IAM-Richtliniensimulator testen. Er stellt automatisch eine Liste der Ressourcen und Parameter bereit, die für jede AWS Aktion erforderlich sind, einschließlich Amazon DocumentDB-Aktionen. Der IAM-Richtliniensimulator bestimmt die Berechtigungen, die für jede der von Ihnen angegebenen Aktionen erforderlich sind. Informationen zum IAM-Richtliniensimulator finden Sie im IAM-Benutzerhandbuch unter Testen von IAM-Richtlinien mit dem IAM-Richtliniensimulator.

> **Note**
>
> Um eine Aktion anzugeben, verwenden Sie das Präfix `rds:` gefolgt vom Namen der API-Operation (z. B. `rds:CreateDBInstance`).

Im Folgenden sind Amazon RDS-API-Operationen und die zugehörigen Aktionen, Ressourcen und Bedingungsschlüssel aufgeführt.

Themen

- Amazon DocumentDB-Aktionen, die Berechtigungen auf Ressourcenebene unterstützen
- Amazon DocumentDB-Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen

Amazon DocumentDB-Aktionen, die Berechtigungen auf Ressourcenebene unterstützen

Resource-level Berechtigungen bieten die Möglichkeit, die Ressourcen anzugeben, für die Benutzer Aktionen ausführen dürfen. Amazon DocumentDB unterstützt teilweise Berechtigungen auf Ressourcenebene. Das bedeutet, dass Sie für bestimmte Amazon DocumentDB-Aktionen anhand von Bedingungen, die erfüllt werden müssen, oder anhand bestimmter Ressourcen, die Benutzer verwenden dürfen, steuern können, wann Benutzer diese Aktionen verwenden dürfen. Beispielsweise können Sie Benutzern die Berechtigung erteilen, nur bestimmte Instances zu ändern.

Im Folgenden sind die Amazon DocumentDB-API-Operationen und die zugehörigen Aktionen, Ressourcen und Bedingungsschlüssel aufgeführt.

> **Note**
>
> Für bestimmte Verwaltungsfunktionen verwendet Amazon DocumentDB Betriebstechnologie, die mit Amazon RDS gemeinsam genutzt wird. Weitere Amazon DocumentDB-Aktionen und -Berechtigungen finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon RDS in der Service Authorization Reference.

Amazon DocumentDB-API-Operationen und -Aktionen	Ressourcen	Bedingungsschlüssel
AddTagsToResource rds:AddTagsToResource	Instance arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
	Subnetzgruppe arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
ApplyPendingMaintenanceAction rds:ApplyPendingMaintenanceAction	Instance arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
CopyDBClusterSnapshot rds:CopyDBClusterSnapshot	Cluster-Snapshot arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag
CreateDBCluster rds:CreateDBCluster	Cluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	rds:cluster-tag
	Cluster-Parametergruppe	rds:cluster-pg-tag

Amazon DocumentDB-API-Operationen und -Aktionen	Ressourcen	Bedingungsschlüssel
	arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	
	Subnetzgruppe arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
CreateDBClusterParameterGroup rds:CreateDBClusterParameterGroup	Cluster-Parametergruppe arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
CreateDBClusterSnapshot rds:CreateDBClusterSnapshot	Cluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	rds:cluster-tag
	Cluster-Snapshot arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag
CreateDBInstance rds:CreateDBInstance	Instance arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:DatabaseClass rds:db-tag

Amazon DocumentDB-API-Operationen und -Aktionen	Ressourcen	Bedingungsschlüssel
	Cluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	rds:cluster-tag
CreateDBSubnetGroup rds:CreateDBSubnetGroup	Subnetzgruppe arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
DeleteDBInstance rds:DeleteDBInstance	Instance arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
DeleteDBSubnetGroup rds:DeleteDBSubnetGroup	Subnetzgruppe arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
DescribeDBClusterParameterGroups rds:DescribeDBClusterParameterGroups	Cluster-Parametergruppe arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag

Amazon DocumentDB-API-Operationen und -Aktionen	Ressourcen	Bedingungsschlüssel
DescribeDBClusterParameters rds:DescribeDBClusterParameters	Cluster-Parametergruppe arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
DescribeDBClusters rds:DescribeDBClusters	Cluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag
DescribeDBClusterSnapshotAttributes rds:DescribeDBClusterSnapshotAttributes	Cluster-Snapshot arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag
DescribeDBSubnetGroups rds:DescribeDBSubnetGroups	Subnetzgruppe arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
DescribePendingMaintenanceActions rds:DescribePendingMaintenanceActions	Instance arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:DatabaseClass rds:db-tag

Amazon DocumentDB-API-Operationen und -Aktionen	Ressourcen	Bedingungsschlüssel
FailoverDBCluster rds:FailoverDBCluster	Cluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag
ListTagsForResource rds:ListTagsForResource	Instance arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
	Subnetzgruppe arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
ModifyDBCluster rds:ModifyDBCluster	Cluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	rds:cluster-tag
	Cluster-Parametergruppe arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
ModifyDBClusterParameterGroup rds:ModifyDBClusterParameterGroup	Cluster-Parametergruppe arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag

Amazon DocumentDB-API-Operationen und -Aktionen	Ressourcen	Bedingungsschlüssel
ModifyDBClusterSnapshotAttribute rds:ModifyDBClusterSnapshotAttribute	Cluster-Snapshot arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag
ModifyDBInstance rds:ModifyDBInstance	Instance arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:DatabaseClass rds:db-tag
RebootDBInstance rds:RebootDBInstance	Instance arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
RemoveTagsFromResource rds:RemoveTagsFromResource	Instance arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
	Subnetzgruppe arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
ResetDBClusterParameterGroup rds:ResetDBClusterParameterGroup	Cluster-Parametergruppe arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag

Amazon DocumentDB-API-Operationen und -Aktionen	Ressourcen	Bedingungsschlüssel
RestoreDBClusterFromSnapshot rds:RestoreDBClusterFromSnapshot	Cluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag
	Cluster-Snapshot arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag
RestoreDBClusterToPointInTime rds:RestoreDBClusterToPointInTime	Cluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag
	Subnetzgruppe arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag

Amazon DocumentDB-Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen

Sie können alle Amazon DocumentDB-Aktionen in einer IAM-Richtlinie verwenden, um Benutzern entweder die Berechtigung zur Verwendung dieser Aktion zu gewähren oder zu verweigern. Allerdings unterstützen nicht alle Amazon DocumentDB-Aktionen Berechtigungen auf Ressourcenebene, mit denen Sie die Ressourcen angeben können, für die eine Aktion ausgeführt werden kann. Die folgenden Amazon DocumentDB-API-Aktionen unterstützen derzeit keine Berechtigungen auf Ressourcenebene. Um diese Aktionen in einer IAM-Richtlinie verwenden zu

können, müssen Sie den Benutzern daher die Erlaubnis gewähren, alle Ressourcen für die Aktion zu verwenden, indem Sie einen * Platzhalter für das Element in Ihrer Anweisung verwenden. `Resource`

- `rds:DescribeDBClusterSnapshots`
- `rds:DescribeDBInstances`

Cross-service verwirrt, stellvertretender Vertreter der Prävention

Das Problem des verwirrten Stellvertreters ist ein Sicherheitsproblem, bei dem eine Entität, die keine Berechtigung zur Durchführung einer Aktion hat, eine privilegiertere Entität zur Durchführung der Aktion zwingen kann. Insofern AWS kann dienststellenübergreifender Identitätswechsel zu einem verwirrten Stellvertreterproblem führen. Cross-service Ein Identitätswechsel kann auftreten, wenn ein Dienst (der aufrufende Dienst) einen anderen Dienst (den aufgerufenen Dienst) anruft. Der aufrufende Service kann manipuliert werden, um seine Berechtigungen zu verwenden, um Aktionen auf die Ressourcen eines anderen Kunden auszuführen, für die er sonst keine Zugriffsberechtigung haben sollte. Um dies zu verhindern, bietet AWS Tools, mit denen Sie Ihre Daten für alle Services mit Serviceprinzipalen schützen können, die Zugriff auf Ressourcen in Ihrem Konto erhalten haben.

Verwenden Sie die Schlüssel `aws:SourceArn` und den `aws:SourceAccount` globalen Bedingungskontext in den Ressourcenrichtlinien, um die Berechtigungen einzuschränken, die Amazon DocumentDB einem anderen Dienst für die Ressource erteilt. Verwenden Sie `aws:SourceArn`, wenn Sie nur eine Ressource mit dem betriebsübergreifenden Zugriff verknüpfen möchten. Verwenden Sie `aws:SourceAccount`, wenn Sie zulassen möchten, dass Ressourcen in diesem Konto mit der betriebsübergreifenden Verwendung verknüpft werden.

Der effektivste Weg, um sich vor dem Confused-Deputy-Problem zu schützen, ist die Verwendung des globalen Bedingungskontext-Schlüssels `aws:SourceArn` mit dem vollständigen ARN der Ressource. Wenn Sie den vollständigen ARN der Ressource nicht kennen oder wenn Sie mehrere Ressourcen angeben, verwenden Sie den globalen Kontextbedingungsschlüssel `aws:SourceArn` mit Platzhalterzeichen (*) für die unbekannten Teile des ARN. Beispiel, `arn:aws:`*`servicename`*`:*:`*`123456789012`*`:*`.

Wenn der `aws:SourceArn`-Wert die Konto-ID nicht enthält, z. B. einen Amazon-S3-Bucket-ARN, müssen Sie beide globale Bedingungskontextschlüssel verwenden, um Berechtigungen einzuschränken.

Der Wert von `aws:SourceArn` muss Cluster sein.

Das folgende Beispiel zeigt, wie Sie die Condition-Context-Schlüssel `aws:SourceArn` und die `aws:SourceAccount` globalen Condition-Context-Schlüssel in Amazon DocumentDB verwenden können, um das verwirrte Deputy-Problem zu vermeiden.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "servicename.amazonaws.com"
    },
    "Action": "servicename:ActionName",
    "Resource": [
      "arn:aws:servicename:::ResourceName/*"
    ],
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:servicename:*:123456789012:*"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

Authentifizierung mithilfe der IAM-Identität

Amazon DocumentDB-Benutzer und -Anwendungen können IAM-Benutzer und -Rollen verwenden, um sich bei einem Amazon DocumentDB-Cluster zu authentifizieren. Die Amazon DocumentDB-IAM-Authentifizierung ist eine kennwortlose Authentifizierungsmethode. Außerdem senden Client-Anwendungen die Passwortgeheimnisse nicht an den Amazon DocumentDB-Cluster, wenn sie IAM verwenden. roles/users Stattdessen werden Client-Verbindungen AWS STS mithilfe temporärer Sicherheitstoken authentifiziert. Non-administrative Benutzer und Anwendungen können jetzt denselben IAM-Identitäts-ARN verwenden, wenn sie eine Verbindung zu verschiedenen Amazon DocumentDB-Clustern und anderen Diensten herstellen. AWS

Sie können auch wählen, ob Sie sowohl die kennwortbasierte als auch die IAM-Authentifizierung verwenden möchten, um Benutzer und Anwendungen bei einem Amazon DocumentDB-Cluster zu

authentifizieren. Die IAM-Authentifizierung ist nur in der instanzbasierten Cluster-Version 5.0 von Amazon DocumentDB verfügbar. Die IAM-Authentifizierung mithilfe von IAM-Identitäts-ARNs wird für den primären Amazon DocumentDB-Benutzer nicht unterstützt.

> **Note**
>
> Der Hauptbenutzer kann nur mithilfe der vorhandenen kennwortbasierten Authentifizierung authentifiziert werden.

Topics

- Erste Schritte mit der Authentifizierung mithilfe von IAM-Benutzern und -Rollen
- Konfigurieren AWS berechnet Typen für die Authentifizierung bei Amazon DocumentDB mit AWS IAM
- Überwachung von IAM-Authentifizierungsanfragen
- Verwenden der IAM-Authentifizierung
- Treiber, die IAM unterstützen
- Häufig gestellte Fragen zur IAM-Identitätsauthentifizierung

Erste Schritte mit der Authentifizierung mithilfe von IAM-Benutzern und -Rollen

Amazon DocumentDB-Benutzer und -Rollen mit IAM-Identitäten werden in einer Datenbank erstellt und verwaltet. `$external`

Einen Benutzer erstellen

Stellen Sie als Hauptbenutzer eine Verbindung her und erstellen Sie dann einen IAM-Benutzer und eine IAM-Rolle:

```
use $external;
db.createUser(
    {
        user: "arn:aws:iam::123456789123:user/iamuser",
        mechanisms: ["MONGODB-AWS"],
        roles: [ { role: "readWrite", db: "readWriteDB" } ]
    }
```

```
);
```

Fügen Sie alternativ einen Amazon DocumentDB-Benutzer mithilfe einer IAM-Rolle hinzu:

```
use $external;
db.createUser(
    {
        user: "arn:aws:iam::123456789123:role/iamrole",
        mechanisms: ["MONGODB-AWS"],
        roles: [ { role: "readWrite", db: "readWriteDB" } ]
    }
);
```

Einen IAM-Benutzer oder eine IAM-Rolle ändern

Ändern Sie einen vorhandenen IAM-Benutzer:

```
use $external;
db.updateUser(
  "arn:aws:iam::123456789123:user/iamuser",
  {
    roles: [ { role: "read", db: "readDB" } ]
  }
);
```

Ändern Sie eine bestehende IAM-Rolle:

```
use $external;
db.updateUser(
  "arn:aws:iam::123456789123:role/iamrole",
  {
    roles: [ { role: "read", db: "readDB" } ]
  }
);
```

So gewähren oder entziehen Sie einem IAM-Benutzer Rollen:

```
use $external;
db.grantRolesToUser(
  "arn:aws:iam::123456789123:user/iamuser",
  [ { db: "admin", role: "readWriteAnyDatabase" } ]
);
```

```
use $external;
db.revokeRolesFromUser(
  "arn:aws:iam::123456789123:user/iamuser",
  [ { db: "admin", role: "readWriteAnyDatabase" } ]
);
```

So gewähren oder entziehen Sie Rollen einer IAM-Rolle:

```
use $external;
db.grantRolesToUser(
  "arn:aws:iam::123456789123:user/iamrole",
  [ { db: "admin", role: "readWriteAnyDatabase" } ]
);
```

```
use $external;
db.revokeRolesFromUser(
  "arn:aws:iam::123456789123:user/iamrole",
  [ { db: "admin", role: "readWriteAnyDatabase" } ]
);
```

Einen IAM-Benutzer oder eine IAM-Rolle löschen

So löschen Sie einen vorhandenen IAM-Benutzer:

```
use $external;
db.dropUser("arn:aws:iam::123456789123:user/iamuser");
```

So löschen Sie eine bestehende IAM-Rolle:

```
use $external;
db.dropUser("arn:aws:iam::123456789123:role/iamrole");
```

Konfigurieren Sie einen Verbindungs-URI für die Authentifizierung mit IAM AWS

Verwenden Sie die folgenden URI-Parameter, um sich mit AWS IAM zu authentifizieren: `authSource` as und as. `$external` `authMechanism` `MONGODB-AWS` Wenn Sie einen IAM-Benutzer verwenden, werden die Felder Benutzername und Passwort durch einen Zugriffsschlüssel bzw. einen geheimen Schlüssel ersetzt. Wenn Sie eine IAM-Rolle übernehmen, die der Umgebung zugeordnet ist, in der Sie sich befinden (z. B. AWS Lambda Funktion, Amazon EC2-Instance). Sie müssen bei der Authentifizierung mithilfe des Mechanismus keine speziellen Anmeldeinformationen

angeben. MONGODB-AWS Wenn Sie MongoDB-Treiber verwenden, die den MONGODB-AWS Authentifizierungsmechanismus unterstützen, können die Treiber auch IAM-Rollenanmeldedaten von der Compute-Instance abrufen (z. B. Amazon EC2, Lambda-Funktion und andere). Das folgende Beispiel verwendet eine Mongo-Shell zur Authentifizierung, MONGODB-AWS indem ein Zugriffsschlüssel und ein geheimer Schlüssel (eines IAM-Benutzers) manuell übergeben werden, um die Authentifizierung gegen Amazon DocumentDB zu demonstrieren.

Im folgenden Beispiel wird Python-Code zur Authentifizierung verwendet, MONGODB-AWS ohne dass explizit Anmeldeinformationen übergeben werden (unter Verwendung einer an die Umgebung angehängten IAM-Rolle), um die Authentifizierung gegenüber Amazon DocumentDB zu demonstrieren.

```
##Create a MongoDB client, open a connection to Amazon DocumentDB using an IAM role
    client = pymongo.MongoClient('mongodb://<DocDBEndpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false&authSource=
%24external&authMechanism=MONGODB-AWS')
```

Das folgende Beispiel verwendet eine Mongo-Shell zur Authentifizierung mithilfe eines MONGODB-AWS Mechanismus, bei dem ein Zugriffsschlüssel und ein geheimer Schlüssel (eines IAM-Benutzers) manuell übergeben werden, um die Authentifizierung gegenüber Amazon DocumentDB zu demonstrieren.

```
$ mongo 'mongodb://<access_key>:<secret_key>@<cluster_endpoint>:<db_port>/test?
authSource=%24external&authMechanism=MONGODB-AWS'
```

Im folgenden Beispiel wird eine Mongo-Shell zur Authentifizierung verwendet, MONGODB-AWS ohne dass Anmeldeinformationen explizit übergeben werden (unter Verwendung der an die Umgebung angehängten IAM-Rolle), um die Authentifizierung gegenüber Amazon DocumentDB zu demonstrieren.

```
$ mongo 'mongodb://<cluster_endpoint>:<db_port>/test?authSource=
%24external&authMechanism=MONGODB-AWS'
```

Konfigurieren AWS berechnet Typen für die Authentifizierung bei Amazon DocumentDB mit AWS IAM

Verwenden von Amazon EC2//AWS LambdaAWS Fargate

Amazon EC2 verwendet die folgenden Umgebungsvariablen. Wenn der EC2-Instance eine IAM-Rolle oder eine Ausführungs-IAM-Rolle zugeordnet ist, die einer Lambda-Funktion oder einer Amazon ECS-Aufgabe zugeordnet ist, werden diese Variablen automatisch aufgefüllt und der Treiber kann diese Werte aus der Umgebung abrufen:

```
AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY
AWS_SESSION_TOKEN
```

Weitere Informationen zu Umgebungsvariablen finden Sie unter Verwenden von Lambda-Umgebungsvariablen im Entwicklerhandbuch. AWS Lambda

Verwenden von Amazon EKS

Wenn Sie Ihren Amazon Elastic Kubernetes Service (Amazon EKS) -Pods eine Rolle zuweisen, werden automatisch die folgenden beiden Umgebungsvariablen eingerichtet:

```
AWS_WEB_IDENTITY_TOKEN_FILE - path of web identity token file
AWS_ROLE_ARN - Name of IAM role to connect with
```

Übernehmen Sie mithilfe dieser Variablen manuell die Rolle aus Ihrem Code mithilfe des AWS SDK-Aufrufs für: `AssumeRoleWithWebIdentity`

- Lassen Sie den `ProviderID` Parameter weg.
- Suchen Sie den Wert des `WebIdentityToken` Parameters in der in der `AWS_WEB_IDENTITY_TOKEN_FILE` Umgebungsvariablen beschriebenen Datei.

Weitere Informationen zu Amazon EKS finden Sie unter Was ist Amazon EKS im Amazon EKS-Benutzerhandbuch.

Überwachung von IAM-Authentifizierungsanfragen

Verwenden von Amazon DocumentDB-Auditing

Gehen Sie in Amazon zum Audit-Log-Ordner und verwenden Sie verschiedene Suchmuster CloudWatch, um die Protokolle für die IAM-Authentifizierung abzurufen. Verwenden Sie es beispielsweise `{ $.param.mechanism = "MONGODB-AWS" }` als Suchmuster für „Alle Log-Streams durchsuchen“.

Weitere Informationen zu unterstützten Ereignissen im Auditing finden Sie unterPrüfung von Amazon DocumentDB-Ereignissen.

Verwenden von CloudWatch Amazon-Metriken

`StsGetCallerIdentityCalls`: Diese Metrik zeigt, wie viele `GetCallerIdentity` Aufrufe eine Amazon DocumentDB-Instance an den regionalized AWS -Security-Token-Service (AWS STS) -Endpunkt tätigt. Lesen Sie in der `MONGODB-AWS` Authentifizierungsspezifikation nach, warum Datenbankinstanzen `GetCallerIdentity` STS-Aufrufe tätigen müssen.

Verwenden der IAM-Authentifizierung

Wenn Sie den Benutzernamen und das Passwort nicht in Ihrer eigenen Datenbank verwalten möchten, können Sie die IAM-Authentifizierung verwenden. Die IAM-Authentifizierung ist nur in der instanzbasierten Cluster-Version 5.0 von Amazon DocumentDB verfügbar.

Die IAM-Authentifizierung ist abhängig von (). AWS -Security-Token-Service AWS STS Wenn bei der Verwendung der IAM-Authentifizierung eine AWS STS Drosselungsausnahme auftritt, senken Sie Ihre Verbindungsrate.

Informationen zu IAM-Kontingenten finden Sie unter IAM und AWS STS Kontingente im IAM-Benutzerhandbuch.

Treiber, die IAM unterstützen

Treiber, die Amazon DocumentDB 5.0 und den `MONGODB-AWS` Authentifizierungsmechanismus unterstützen, sollten mit der IAM-Authentifizierungsimplementierung in Amazon DocumentDB funktionieren.

> **Important**
>
> Es gibt eine bekannte Einschränkung bei Node.js Treibern, die älter als Version 6.13.1 sind und die derzeit von der IAM-Identitätsauthentifizierung für Amazon DocumentDB nicht unterstützt werden. Node.js Treiber und Tools, die Node.js Treiber verwenden (z. B. Mongosh), müssen aktualisiert werden, um die Node.js Treiberversion 6.13.1 oder höher zu verwenden.

Häufig gestellte Fragen zur IAM-Identitätsauthentifizierung

Gibt es Beispiele, auf die ich mich beziehen kann?

Auf diesen Seiten finden Sie Beispiele für Anwendungsfälle und Konfigurationen:

- Wie sich menschliche Benutzer mithilfe von IAM-Benutzern und IAM-Rollen bei Amazon DocumentDB authentifizieren können
- Password-less Authentifizierung bei Amazon DocumentDB mithilfe von IAM-Rollen

Ich erhalte eine Fehlermeldung bei der Verwendung meines Python-Treibers: „pymongo.errors. ConfigurationError: Die AWS MONGODB-Authentifizierung erfordert „pymongo-auth-aws“. Wie kann ich das lösen?

Stellen Sie sicher, dass Sie bei der Installation des Python-Treibers mit IAM-Authentifizierung die folgende Anweisung verwenden:

```
pip install 'pymongo[aws]'
```

Dadurch werden die zusätzlichen AWS Abhängigkeiten installiert, die erforderlich sind, damit die IAM-Authentifizierung funktioniert.

Wird meine Verbindung unterbrochen, wenn meine temporären Anmeldeinformationen für die IAM-Rolle ablaufen?

Nein, die temporären IAM-Anmeldeinformationen werden nur für den Verbindungsaufbau und die Authentifizierung verwendet. Dann erfolgt die gesamte weitere Autorisierung im Amazon DocumentDB-Cluster. Selbst wenn es sich um IAM-Anmeldeinformationen handelt rotate/expire, wird die Verbindung nicht unterbrochen oder veraltet.

Authentifizierung mit SCRAM-SHA-256

Amazon DocumentDB (mit MongoDB-Kompatibilität) unterstützt den Salted Challenge Response Authentication Mechanism (SCRAM) für die kennwortbasierte Authentifizierung. Mit SCRAM überprüft der Server, ob ein Client das Passwort eines Benutzers kennt, ohne das Passwort an den Server zu übertragen. Amazon DocumentDB unterstützt zwei SCRAM-Mechanismen: und. `SCRAM-SHA-1` `SCRAM-SHA-256` `SCRAM-SHA-256`verwendet die stärkere SHA-256 Hash-Funktion und arbeitet mit dem FIPS-Modus.

Sie weisen einem Benutzer einen oder beide SCRAM-Mechanismen zu, wenn Sie diesen Benutzer erstellen oder aktualisieren. Benutzer, die mit erstellt wurden, arbeiten ohne Änderungen `SCRAM-SHA-1` weiter. SCRAM-SHA-256 Die Authentifizierung ist in den auf Amazon DocumentDB-Instances basierenden Cluster-Nebenversionen 5.0.1+ und 8.0.1+ verfügbar.

> **SCRAM- und IAM-Authentifizierung schließen sich gegenseitig aus**
>
> Ein Benutzer kann entweder einen SCRAM-Mechanismus oder den `MONGODB-AWS` Mechanismus verwenden, aber nicht beide. Informationen zur IAM-Identitätsauthentifizierung finden Sie unter. Authentifizierung mithilfe der IAM-Identität

Topics

- Benutzer mit SCRAM-Mechanismen erstellen
- Bestehende Benutzer aktualisieren
- Authentifizieren mit SCRAM-SHA-256

Benutzer mit SCRAM-Mechanismen erstellen

Wenn Sie einen Benutzer erstellen, können Sie die SCRAM-Mechanismen angeben, die mit dem `mechanisms` Feld aktiviert werden sollen. In den folgenden Beispielen werden Benutzer mit `SCRAM-SHA-1` nur, `SCRAM-SHA-256` nur und beiden Mechanismen erstellt:

```
db.runCommand({
    createUser: 'sha1user',
    pwd: passwordPrompt(),
    roles: ['readWrite'],
    mechanisms: ['SCRAM-SHA-1']
});
db.runCommand({
    createUser: 'sha256user',
    pwd: passwordPrompt(),
    roles: ['readWrite'],
    mechanisms: ['SCRAM-SHA-256']
});
db.runCommand({
    createUser: 'bothuser',
    pwd: passwordPrompt(),
```

```
    roles: ['readWrite'],
    mechanisms: ['SCRAM-SHA-1','SCRAM-SHA-256']
});
```

Standardmechanismen

Wenn Sie das `mechanisms` Feld nicht angeben, aktiviert Amazon DocumentDB beides `SCRAM-SHA-1` und `SCRAM-SHA-256` für den Benutzer. Die folgenden beiden Befehle sind gleichwertig:

```
db.runCommand({
    createUser: 'user1',
    pwd: passwordPrompt(),
    roles: ['readWrite']
});
// is equivalent to:
db.runCommand({
    createUser: 'user1',
    pwd: passwordPrompt(),
    roles: ['readWrite'],
    mechanisms: ['SCRAM-SHA-1','SCRAM-SHA-256']
});
```

Wenn Sie beide Mechanismen standardmäßig aktivieren, bleibt die Kompatibilität mit älteren MongoDB-Treibern erhalten, die nur `SCRAM-SHA-1` Unterstützung bieten. Neuere Treiber und der FIPS-Modus werden bevorzugt. `SCRAM-SHA-256` Wenn Sie nicht möchten, dass sich ein Benutzer authentifiziert`SCRAM-SHA-1`, geben Sie nur `SCRAM-SHA-256` im `mechanisms` Feld für alle Ihre Benutzer an.

Bestehende Benutzer aktualisieren

Mit erstellte Benutzer werden `SCRAM-SHA-1` weiterhin ohne Änderungen unterstützt. Um einen bestehenden Benutzer von `SCRAM-SHA-1` zu zu wechseln`SCRAM-SHA-256`, aktualisieren Sie den Benutzer mit einem neuen Passwort und gehen Sie wie folgt `SCRAM-SHA-256` vor:

```
db.runCommand({
    updateUser: 'existing',
    pwd: passwordPrompt(),
    mechanisms: ['SCRAM-SHA-256']
});
```

Authentifizieren mit SCRAM-SHA-256

Ihr Treiber authentifiziert sich automatisch anhand der Parameter der Verbindungszeichenfolge, bevor er bei jeder Verbindung den ersten Befehl sendet. `SCRAM-SHA-256`Um einen Verbindungs-URI auszuwählen, setzen Sie den `authMechanism` Parameter auf`SCRAM-SHA-256`:

```
mongodb://username:password@hostname:port/authenticationDatabase?authMechanism=SCRAM-SHA-256
```

Sie können auch eine explizite Authentifizierungsanfrage von Mongosh aus senden:

```
db.auth({user: 'sha256user', pwd: passwordPrompt(), mechanism: 'SCRAM-SHA-256'})
Enter password:
1
```

Wenn Sie in Ihren Verbindungsoptionen keinen Authentifizierungsmechanismus angeben, wählt der Treiber automatisch einen Mechanismus aus, den der Zieldatenbankbenutzer unterstützt.

Verwaltung von Amazon DocumentDB-Benutzern

In Amazon DocumentDB authentifizieren sich Benutzer bei einem Cluster in Verbindung mit einem Passwort. Jeder Cluster verfügt über primäre Anmeldeinformationen, die bei der Clustererstellung festgelegt werden.

> **Note**
>
> Allen neuen Benutzern, die vor dem 26. März 2020 erstellt wurden, wurden die `dbAdminAnyDatabase`-, `readWriteAnyDatabase`- und `clusterAdmin`-Rollen erteilt. Es wird empfohlen, alle Benutzer neu zu bewerten und die Rollen nach Bedarf zu ändern, um die geringstmöglichen Berechtigungen für alle Benutzer in Ihren Clustern zu erzwingen. Weitere Informationen finden Sie unter Datenbankzugriff mit Role-Based Access Control.

Primär- und **`ServiceAdmin`-Benutzer**

Ein neu erstellter Amazon DocumentDB-Cluster hat zwei Benutzer: den Hauptbenutzer und den `serviceadmin` Benutzer.

Der Hauptbenutzer ist ein einzelner, privilegierter Benutzer, der Verwaltungsaufgaben ausführen und zusätzliche Benutzer mit Rollen erstellen kann. Wenn Sie zum ersten Mal eine Verbindung zu einem Amazon DocumentDB-Cluster herstellen, müssen Sie sich mit den primären Anmeldeinformationen authentifizieren. Der Hauptbenutzer erhält diese Administratorberechtigungen für einen Amazon DocumentDB-Cluster, wenn dieser Cluster erstellt wird, und ihm wird die Rolle von zugewiesen. `root`

Der `serviceadmin`-Benutzer wird implizit beim Erstellen des Clusters erstellt. Jeder Amazon DocumentDB-Cluster hat einen `serviceadmin` Benutzer, AWS der Ihnen die Möglichkeit bietet, Ihren Cluster zu verwalten. Sie können sich mit `serviceadmin` nicht anmelden, den Benutzer löschen oder umbenennen, sein Passwort ändern oder die Berechtigungen ändern. Jeder Versuch, das zu tun, führt zu einem Fehler.

> **Note**
>
> Der primäre `serviceadmin` Benutzer und die Benutzer für einen Amazon DocumentDB-Cluster können nicht gelöscht werden, und die Rolle von `root` für den primären Benutzer kann nicht entzogen werden.
> Wenn Sie Ihr primäres Benutzerkennwort vergessen haben, können Sie es mit AWS-Managementkonsole oder dem zurücksetzen. AWS CLI

Zusätzliche Benutzer erstellen

Nachdem Sie eine Verbindung als Hauptbenutzer (oder als beliebiger Benutzer, der die Rolle hat`createUser`) hergestellt haben, können Sie einen neuen Benutzer erstellen, wie unten gezeigt.

```
db.createUser(
    {
        user: "sample-user-1",
        pwd: "password123",
        roles:
            [{"db":"admin", "role":"dbAdminAnyDatabase" }]
    }
)
```

Um Benutzerdetails anzuzeigen, können Sie den Befehl `show users` wie folgt verwenden. Sie können Benutzer zusätzlich mit dem Befehl `dropUser` entfernen. Weitere Informationen finden Sie unter Allgemeine Befehle.

```
show users
{
    "_id" : "serviceadmin",
    "user" : "serviceadmin",
    "db" : "admin",
    "roles" : [
     {
            "role" : "root",
            "db" : "admin"
        }
    ]
},

{
    "_id" : "myPrimaryUser",
    "user" : "myPrimaryUser",
    "db" : "admin",
    "roles" : [
     {
            "role" : "root",
            "db" : "admin"
        }
    ]
},

{
    "_id" : "sample-user-1",
    "user" : "sample-user-1",
    "db" : "admin",
    "roles" : [
     {
            "role" : "dbAdminAnyDatabase",
            "db" : "admin"
     }
    ]
}
```

Im obigen Beispiel wird der neue Benutzer `sample-user-1` der `admin`-Datenbank zugewiesen. Dies ist bei einem neuen Benutzer immer der Fall. Amazon DocumentDB hat nicht das Konzept einer `authenticationDatabase` und daher erfolgt die gesamte Authentifizierung im Kontext der `admin` Datenbank.

Wenn Sie beim Erstellen von Benutzern das db Feld bei der Angabe der Rolle weglassen, ordnet Amazon DocumentDB die Rolle implizit der Datenbank zu, für die die Verbindung hergestellt wird. Wenn Ihre Verbindung beispielsweise für die Datenbank `sample-database` hergestellt wird und Sie den folgenden Befehl ausführen, wird der Benutzer `sample-user-2` in der Datenbank `admin` erstellt und verfügt über `readWrite`-Berechtigungen für die Datenbank `sample-database`.

```
db.createUser(
    {
        user: "sample-user-2",
        pwd: "password123",
        roles:
            ["readWrite"]
    }
)
```

Wenn Sie Benutzer mit Rollen erstellen, die über alle Datenbanken hinweg erfasst sind (z. B. `readInAnyDatabase`), müssen Sie sich beim Erstellen des Benutzers entweder im Kontext der `admin`-Datenbank befinden oder beim Erstellen des Benutzers explizit die Datenbank für die Rolle angeben.

Um den Kontext Ihrer Datenbank zu wechseln, können Sie den folgenden Befehl verwenden.

```
use admin
```

Weitere Informationen zur rollenbasierten Zugriffssteuerung und zum Erzwingen geringstmöglicher Berechtigungen unter den Benutzern im Cluster finden Sie unter Datenbankzugriff mit Role-Based Access Control.

Automatisch rotierende Passwörter für Amazon DocumentDB

Mit AWS Secrets Manager können Sie fest codierte Anmeldeinformationen in Ihrem Code (einschließlich Passwörter) durch einen API-Aufruf an Secrets Manager ersetzen, um das Geheimnis programmgesteuert abzurufen. Dadurch wird sichergestellt, dass das Secret nicht kompromittiert

werden kann, wenn jemand Ihren Code durchsucht, da es sich gar nicht dort befindet. Außerdem können Sie Secrets Manager so konfigurieren, dass er das Secret automatisch nach einem von Ihnen festgelegten Zeitplan rotiert. So können Sie Secrets mit langer Einsatzdauer durch Secrets mit kurzer Einsatzdauer ersetzen und damit das Risiko einer Kompromittierung erheblich verringern.

Mit Secrets Manager können Sie Ihre Amazon DocumentDB-Passwörter (d. h. Secrets) mithilfe einer AWS Lambda Funktion, die Secrets Manager bereitstellt, automatisch rotieren.

Weitere Informationen über Amazon DocumentDB AWS Secrets Manager und die native Integration mit Amazon DocumentDB finden Sie unter den folgenden Links:

- Blog: Wie rotiere ich Amazon DocumentDB- und Amazon Redshift-Anmeldeinformationen in Secrets Manager AWS
- Was ist AWS Secrets Manager?
- AWS Secrets Manager Geheimnisse rotieren
- Amazon DocumentDB-Anmeldeinformationen im Secrets Manager

Datenbankzugriff mit Role-Based Access Control

Mithilfe der rollenbasierten Zugriffskontrolle (RBAC) in Amazon DocumentDB (mit MongoDB-Kompatibilität) können Sie den Zugriff auf die Aktionen einschränken, die Benutzer an Datenbanken ausführen können. Bei RBAC werden einem Benutzer eine oder mehrere Rollen gewährt. Diese Rollen bestimmen die Operationen, die ein Benutzer mit Datenbankressourcen ausführen kann. Amazon DocumentDB unterstützt derzeit sowohl integrierte Rollen, die auf Datenbankebene beschränkt sind, wie,,,,,`read`,, `readWritereadAnyDatabase`, als auch benutzerdefinierte Rollen`clusterAdmin`, die auf bestimmte Aktionen beschränkt werden können, und granulare Ressourcen wie Sammlungen, die Ihren Anforderungen entsprechen.

Zu den häufigsten Anwendungsfällen für RBAC gehören die Durchsetzung der geringsten Rechte, indem Benutzer mit schreibgeschütztem Zugriff auf die Datenbanken oder Sammlungen in einem Cluster erstellt werden, und mehrinstanzenfähige Anwendungsdesigns, die es einem einzelnen Benutzer ermöglichen, auf eine bestimmte Datenbank oder Sammlung in einem Cluster zuzugreifen.

> **Note**
>
> Allen neuen Benutzern, die vor dem 26. März 2020 erstellt wurden, wurden die `dbAdminAnyDatabase`-, `readWriteAnyDatabase`- und `clusterAdmin`-Rollen erteilt. Es

wird empfohlen, alle vorhandenen Benutzer neu zu bewerten und die Rollen nach Bedarf zu ändern, um die geringstmöglichen Berechtigungen für Ihre Cluster zu erzwingen.

Themen

- RBAC-Konzepte
- Erste Schritte mit den integrierten RBAC-Rollen
- Erste Schritte mit benutzerdefinierten RBAC-Rollen
- Stellen Sie als Benutzer eine Verbindung zu Amazon DocumentDB her
- Allgemeine Befehle
- Funktionale Unterschiede
- Einschränkungen
- Datenbankzugriff mit Role-Based Access Control

RBAC-Konzepte

Im Folgenden finden Sie wichtige Begriffe und Konzepte zur rollenbasierten Zugriffssteuerung. Weitere Informationen zu Amazon DocumentDB-Benutzern finden Sie unter. Verwaltung von Amazon DocumentDB-Benutzern

- Benutzer — Eine individuelle Entität, die sich bei der Datenbank authentifizieren und Operationen ausführen kann.
- Passwort — Ein Geheimnis, das zur Authentifizierung des Benutzers verwendet wird.
- Rolle — Autorisiert einen Benutzer, Aktionen an einer oder mehreren Datenbanken auszuführen.
- Admin-Datenbank — Die Datenbank, in der Benutzer gespeichert und für die Benutzer autorisiert werden.
- Datenbank (**db**) — Der Namespace innerhalb von Clustern, der Sammlungen zum Speichern von Dokumenten enthält.

Der folgende Befehl erstellt einen Benutzer mit Namen `sample-user`.

```
db.createUser({user: "sample-user", pwd: "abc123", roles: [{role: "read", db: "sample-database"}]})
```

In diesem Beispiel:

- `user: "sample-user"`— Gibt den Benutzernamen an.
- `pwd: "abc123"`— Zeigt das Benutzerkennwort an.
- `role: "read", "db: "sample-database"`— Zeigt an, dass der Benutzer Leseberechtigungen haben `sample-user` wird`sample-database`.



Das folgende Beispiel zeigt die Ausgabe, nachdem Sie den Benutzer `sample-user` mit `db.getUser(sample-user)` erhalten. In diesem Beispiel befindet sich der Benutzer `sample-user` in der Datenbank `admin`, besitzt jedoch die „read“-Rolle für die Datenbank `sample-database`.



Wenn Sie beim Erstellen von Benutzern das db Feld bei der Angabe der Rolle weglassen, ordnet Amazon DocumentDB die Rolle implizit der Datenbank zu, für die die Verbindung hergestellt wird. Wenn Ihre Verbindung beispielsweise für die Datenbank `sample-database` hergestellt wird und Sie den folgenden Befehl ausführen, wird der Benutzer `sample-user` in der Datenbank `admin` erstellt und verfügt über `readWrite`-Berechtigungen für die Datenbank `sample-database`.

```
db.createUser({user: "sample-user", pwd: "abc123", roles: ["readWrite"]})
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
   "user":"sample-user",
```

```
   "roles":[
      {
         "db":"sample-database",
         "role":"readWrite"
      }
   ]
}
```

Wenn Sie Benutzer mit Rollen erstellen, die über alle Datenbanken hinweg erfasst sind (z. B. `readAnyDatabase`), müssen Sie sich beim Erstellen des Benutzers entweder im Kontext der Datenbank `admin` befinden oder beim Erstellen des Benutzers explizit die Datenbank für die Rolle angeben. Um Befehle für die Datenbank `admin` auszugeben, können Sie den Befehl `use admin` verwenden. Weitere Informationen finden Sie unter Allgemeine Befehle.

Erste Schritte mit den integrierten RBAC-Rollen

Um Ihnen den Einstieg in die rollenbasierte Zugriffssteuerung zu erleichtern, führen Sie in diesem Abschnitt ein Beispielszenario zum Erzwingen der geringstmöglichen Berechtigungen durch das Erstellen von Rollen für drei Benutzer mit unterschiedlichen Auftragsfunktionen durch.

- `user1` ist ein neuer Manager, der in der Lage sein muss, alle Datenbanken in einem Cluster anzuzeigen und darauf zuzugreifen.
- `user2` ist ein neuer Mitarbeiter, der Zugriff auf nur eine Datenbank, `sample-database-1`, in demselben Cluster benötigt.
- `user3` ist ein vorhandener Mitarbeiter, der im selben Cluster eine andere Datenbank, `sample-database-2`, anzeigen und darauf zugreifen muss, auf die er vorher keinen Zugriff hatten.

Zu einem späteren Zeitpunkt verlassen sowohl `user1` als auch `user2` das Unternehmen, so dass ihr Zugang widerrufen werden muss.

Um Benutzer zu erstellen und Rollen zu erteilen, muss der Benutzer, mit dem Sie sich beim Cluster authentifizieren, über eine zugeordnete Rolle verfügen, die Aktionen für `createUser` und `grantRole` ausführen kann. Zum Beispiel können die Rollen `admin` und `userAdminAnyDatabase` beide solche Fähigkeiten gewähren. Informationen zu Aktionen pro Rolle finden Sie unter Datenbankzugriff mit Role-Based Access Control.

> **Note**
>
> In Amazon DocumentDB werden alle Benutzer- und Rollenoperationen (z. B.`create`,`get`, `drop` `grantrevoke`, usw.) implizit in der `admin` Datenbank ausgeführt, unabhängig davon, ob Sie Befehle für die Datenbank ausgeben oder nicht. `admin`

Um zunächst zu verstehen, was die aktuellen Benutzer und Rollen im Cluster sind, können Sie den Befehl `show users` ausführen, wie im folgenden Beispiel. Sie sehen zwei Benutzer `serviceadmin` und den Hauptbenutzer für den Cluster. Diese beiden Benutzer sind immer vorhanden und können nicht gelöscht werden. Weitere Informationen finden Sie unter Verwaltung von Amazon DocumentDB-Benutzern.

```
show users
```

Erstellen Sie für `user1` eine Rolle mit Lese- und Schreibzugriff auf alle Datenbanken im gesamten Cluster mit dem folgenden Befehl.

```
db.createUser({user: "user1", pwd: "abc123", roles: [{role: "readWriteAnyDatabase", db:
 "admin"}]})
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
   "user":"user1",
   "roles":[
      {
         "role":"readWriteAnyDatabase",
         "db":"admin"
      }
   ]
}
```

Erstellen Sie für `user2` eine Rolle mit schreibgeschütztem Zugriff auf die Datenbank `sample-database-1` mit dem folgenden Befehl.

```
db.createUser({user: "user2", pwd: "abc123", roles: [{role: "read", db: "sample-
database-1"}]})
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
   "user":"user2",
   "roles":[
      {
         "role":"read",
         "db":"sample-database-1"
      }
   ]
}
```

Um das Szenario zu simulieren, dass `user3` ein vorhandener Benutzer ist, erstellen Sie zuerst den Benutzer `user3` und weisen Sie dann `user3` eine neue Rolle zu.

```
db.createUser({user: "user3", pwd: "abc123", roles: [{role: "readWrite", db: "sample-
database-1"}]})
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
   "user":"user3",
   "roles":[
      {
         "role":"readWrite",
         "db":"sample-database-1"
      }
   ]
}
```

Nachdem der Benutzer `user3` erstellt wurde, weisen Sie `user3` die `read`-Rolle für `sample-database-2` zu.

```
db.grantRolesToUser("user3", [{role: "read", db: "sample-database-2"}])
```

Schließlich verlassen sowohl `user1` als auch `user2` das Unternehmen und ihr Zugriff auf den Cluster muss widerrufen werden. Sie können dies tun, indem Sie die Benutzer wie folgt löschen.

```
db.dropUser("user1")
db.dropUser("user2")
```

Um sicherzustellen, dass alle Benutzer über die entsprechenden Rollen verfügen, können Sie alle Benutzer mit dem folgenden Befehl auflisten.

```
show users
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
   "_id":"serviceadmin",
   "user":"serviceadmin",
   "db":"admin",
   "roles":[
      {
         "db":"admin",
         "role":"root"
      }
   ]
}
{
   "_id":"master-user",
   "user":"master-user",
   "db":"admin",
   "roles":[
      {
         "db":"admin",
         "role":"root"
      }
   ]
}
{
   "_id":"user3",
   "user":"user3",
   "db":"admin",
   "roles":[
      {
         "db":"sample-database-2",
         "role":"read"
      },
      {
         "db":"sample-database-1",
         "role":"readWrite"
      }
   ]
```

```
}
```

Erste Schritte mit benutzerdefinierten RBAC-Rollen

Um Ihnen den Einstieg in benutzerdefinierte Rollen zu erleichtern, werden Sie in diesem Abschnitt durch ein Beispielszenario geführt, in dem Sie die geringsten Rechte erzwingen, indem Rollen für drei Benutzer mit unterschiedlichen Aufgabenbereichen erstellt werden.

In diesem Beispiel gilt Folgendes:

- `user1` ist ein neuer Manager, der in der Lage sein muss, alle Datenbanken in einem Cluster anzuzeigen und darauf zuzugreifen.
- `user2`ist ein neuer Mitarbeiter, der nur die Aktion „Suchen“ für nur eine Datenbank im selben Cluster benötigt. `sample-database-1`
- `user3`ist ein vorhandener Mitarbeiter, der eine bestimmte Sammlung, col2, in einer anderen Datenbank, auf `sample-database-2` die er zuvor keinen Zugriff hatte, im selben Cluster einsehen und darauf zugreifen muss.
- Erstellen Sie für `user1` eine Rolle mit Lese- und Schreibzugriff auf alle Datenbanken im gesamten Cluster mit dem folgenden Befehl.

```
db.createUser(
 {
     user: "user1", pwd: "abc123",
     roles: [{role: "readWriteAnyDatabase", db: "admin"}]
 }
 )
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
   "user":"user1",
   "roles":[
      {
         "role":"readWriteAnyDatabase",
         "db":"admin"
      }
   ]
}
```

Erstellen Sie `user2` z. B. mit dem folgenden Befehl eine Rolle mit Suchrechten für alle Sammlungen in der Datenbank`sample-database-1`. Beachten Sie, dass diese Rolle sicherstellen würde, dass alle zugehörigen Benutzer nur Suchabfragen ausführen können.

```
db.createRole(
{
    role: "findRole",
    privileges: [
    {
        resource: {db: "sample-database-1", collection: ""}, actions: ["find"]
    }],
    roles: []
}
)
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
   "role":"findRole",
   "privileges":[
      {
         "resource":{
            "db":"sample-database-1",
            "collection":""
         },
         "actions":[
            "find"
         ]
      }
   ],
   "roles":[

   ]
}
```

Erstellen Sie als Nächstes den Benutzer (`user2`) und hängen Sie die kürzlich erstellte Rolle `findRole` an den Benutzer an.

```
db.createUser(
{
    user: "user2",
    pwd: "abc123",
```

```
    roles: []
})

 db.grantRolesToUser("user2",["findRole"])
```

Um das Szenario zu simulieren, bei dem es sich um einen vorhandenen Benutzer `user3` handelt`user3`, erstellen Sie zuerst den Benutzer und dann eine neue Rolle namens CollectionRole, die wir im nächsten Schritt zuweisen werden. `user3`

Jetzt können Sie dem eine neue Rolle zuweisen. `user3` Diese neue Rolle ermöglicht es `user3` Ihnen, eine bestimmte Sammlung, Spalte 2, einzufügen, zu aktualisieren, zu löschen und Zugriff auf diese zu finden. `sample-database-2`

```
db.createUser(
{
    user: "user3",
    pwd: "abc123",
    roles: []
})

db.createRole(
{
    role: "collectionRole",
    privileges: [
    {
        resource: {db: "sample-database-2", collection: "col2"}, actions: ["find",
 "update", "insert", "remove"]
    }],
    roles: []
}
)
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
   "role":"collectionRole",
   "privileges":[
      {
         "resource":{
            "db":"sample-database-2",
            "collection":"col2"
         },
```

```
         "actions":[
            "find",
            "update",
            "insert",
            "remove"
         ]
      }
   ],
   "roles":[

   ]
}
```

Nachdem der Benutzer erstellt `user3` wurde, können Sie ihm `user3` die Rolle `collectionFind` zuweisen.

```
db.grantRolesToUser("user3",["collectionRole"])
```

Schließlich verlassen sowohl `user1` als auch `user2` das Unternehmen und ihr Zugriff auf den Cluster muss widerrufen werden. Sie können dies tun, indem Sie die Benutzer wie folgt löschen.

```
db.dropUser("user1")
db.dropUser("user2")
```

Um sicherzustellen, dass alle Benutzer über die entsprechenden Rollen verfügen, können Sie alle Benutzer mit dem folgenden Befehl auflisten.

```
show users
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
   "_id":"serviceadmin",
   "user":"serviceadmin",
   "db":"admin",
   "roles":[
      {
         "db":"admin",
         "role":"root"
      }
```

```
   ]
}
{
   "_id":"master-user",
   "user":"master-user",
   "db":"admin",
   "roles":[
      {
         "db":"admin",
         "role":"root"
      }
   ]
}
{
   "_id":"user3",
   "user":"user3",
   "db":"admin",
   "roles":[
      {
         "db":"admin",
         "role":"collectionRole"
      }
   ]
}
```

Stellen Sie als Benutzer eine Verbindung zu Amazon DocumentDB her

Wenn Sie eine Verbindung zu einem Amazon DocumentDB-Cluster herstellen, stellen Sie eine Verbindung im Kontext einer bestimmten Datenbank her. Wenn Sie in der Verbindungszeichenfolge keine Datenbank angeben, werden Sie standardmäßig automatisch im Kontext der Datenbank `test` mit dem Cluster verbunden. Alle Befehle auf Sammlungsebene wie `insert` und `find` werden für Sammlungen in der Datenbank `test` ausgegeben.

Verwenden Sie den db Befehl in der Mongo-Shell wie folgt, um zu sehen, in welcher Datenbank Sie sich gerade befinden oder für die Sie Befehle ausgeben.

Abfrage:

```
db
```

Ausgabe:

```
test
```

Obwohl sich die Standardverbindung möglicherweise im Kontext der Datenbank `test` befindet, bedeutet dies nicht unbedingt, dass der Benutzer, der der Verbindung zugeordnet ist, berechtigt ist, Aktionen für die Datenbank `test` auszuführen. Wenn Sie sich im vorangegangenen Beispielszenario als Benutzer `user3` authentifizieren, der die `readWrite`-Rolle für die Datenbank `sample-database-1` besitzt, ist der Standardkontext Ihrer Verbindung die Datenbank `test`. Wenn Sie jedoch versuchen, ein Dokument in eine Sammlung in der Datenbank `test` einzufügen, erhalten Sie die Fehlermeldung Autorisierungsfehler. Dies liegt daran, dass dieser Benutzer nicht berechtigt ist, diesen Befehl in dieser Datenbank auszuführen, wie unten gezeigt.

Abfrage:

```
db
```

Ausgabe:

```
test
```

Abfrage:

```
db.col.insert({x:1})
```

Ausgabe:

```
WriteCommandError({ "ok" : 0, "code" : 13, "errmsg" : "Authorization failure" })
```

Wenn Sie den Kontext Ihrer Verbindung zur Datenbank `sample-database-1` ändern, können Sie in die Sammlung schreiben, für die der Benutzer die Berechtigung dazu hat.

Abfrage:

```
use sample-database-1
```

Ausgabe:

```
switched to db sample-database-1
```

Abfrage:

```
db.col.insert({x:1})
```

Ausgabe:

```
WriteResult({ "nInserted" : 1})
```

Wenn Sie sich bei einem Cluster mit einem bestimmten Benutzer authentifizieren, können Sie die Datenbank auch in der Verbindungszeichenfolge angeben. Dadurch entfällt die Notwendigkeit, den Befehl `use` auszuführen, nachdem der Benutzer in der Datenbank `admin` authentifiziert wurde.

Die folgende Verbindungszeichenfolge authentifiziert den Benutzer für die Datenbank `admin`, aber der Kontext der Verbindung wird für die Datenbank `sample-database-1` verwendet.

```
mongo "mongodb://user3:abc123@sample-cluster.node.us-east-1.docdb.amazonaws.com:27017/
sample-database-2"
```

Allgemeine Befehle

Dieser Abschnitt enthält Beispiele für gängige Befehle, die die rollenbasierte Zugriffskontrolle in Amazon DocumentDB verwenden. Sie müssen sich im Kontext der Datenbank `admin` befinden, um Benutzer und Rollen zu erstellen und zu ändern. Sie können den Befehl `use admin` verwenden, um zur Datenbank `admin` zu wechseln.

> **Note**
>
> Änderungen an den Benutzern und Rollen erfolgen implizit in der Datenbank `admin`. Wenn Sie Benutzer mit Rollen erstellen, die über alle Datenbanken hinweg erfasst sind (z. B. `readAnyDatabase`), müssen Sie sich beim Erstellen des Benutzers entweder im Kontext der Datenbank `admin` befinden (d. h. `use admin`) oder beim Erstellen des Benutzers explizit die Datenbank für die Rolle angeben (wie in Beispiel 2 in diesem Abschnitt gezeigt).

Beispiel 1: Erstellen Sie einen Benutzer mit einer `read` Rolle für die Datenbank. `foo`

```
db.createUser({user: "readInFooBar", pwd: "abc123", roles: [{role: "read", db:
 "foo"}]})
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
   "user":"readInFooBar",
   "roles":[
      {
         "role":"read",
         "db":"foo"
      }
   ]
}
```

Beispiel 2: Erstellen Sie einen Benutzer mit Lesezugriff auf alle Datenbanken.

```
db.createUser({user: "readAllDBs", pwd: "abc123", roles: [{role: "readAnyDatabase", db:
 "admin"}]})
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
   "user":"readAllDBs",
   "roles":[
      {
         "role":"readAnyDatabase",
         "db":"admin"
      }
   ]
}
```

Beispiel 3: Gewähren Sie einem vorhandenen Benutzer eine `read` Rolle in einer neuen Datenbank.

```
db.grantRolesToUser("readInFooBar", [{role: "read", db: "bar"}])
```

Beispiel 4: Aktualisieren Sie die Rolle eines Benutzers.

```
db.updateUser("readInFooBar", {roles: [{role: "read", db: "foo"}, {role: "read", db:
 "baz"}]})
```

Beispiel 5: Widerrufen Sie einem Benutzer den Zugriff auf eine Datenbank.

```
db.revokeRolesFromUser("readInFooBar", [{role: "read", db: "baz"}])
```

Beispiel 6: Beschreiben Sie eine integrierte Rolle.

```
db.getRole("read", {showPrivileges:true})
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
   "role":"read",
   "db":"sample-database-1",
   "isBuiltin":true,
   "roles":[

   ],
   "inheritedRoles":[

   ],
   "privileges":[
      {
         "resource":{
            "db":"sample-database-1",
            "collection":""
         },
         "actions":[
            "changeStream",
            "collStats",
            "dbStats",
            "find",
            "killCursors",
            "listCollections",
            "listIndexes"
         ]
      }
   ],
   "inheritedPrivileges":[
      {
         "resource":{
            "db":"sample-database-1",
            "collection":""
         },
         "actions":[
            "changeStream",
            "collStats",
            "dbStats",
```

```
            "find",
            "killCursors",
            "listCollections",
            "listIndexes"
         ]
      }
   }
```

Beispiel 7: Löschen Sie einen Benutzer aus dem Cluster.

```
db.dropUser("readInFooBar")
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
true
```

Beispiel 8: Erstellen Sie eine Rolle mit Lese- und Schreibzugriff auf eine bestimmte Sammlung

```
db.createRole(
{
    role: "collectionRole",
    privileges: [
    {
        resource: {db: "sample-database-2", collection: "col2"}, actions: ["find",
 "update", "insert", "remove"]
    }],
    roles: []
}
)
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
   "role":"collectionRole",
   "privileges":[
      {
         "resource":{
            "db":"sample-database-2",
            "collection":"col2"
         },
         "actions":[
            "find",
```

```
            "update",
            "insert",
            "remove"
         ]
      }
   ],
   "roles":[

   ]
}
```

Beispiel 9: Erstellen Sie einen Benutzer und weisen Sie ihm eine benutzerdefinierte Rolle zu

```
db.createUser(
{
    user: "user3",
    pwd: "abc123",
    roles: []
})

db.grantRolesToUser("user3",["collectionRole"])
```

Beispiel 10: Gewähren Sie einer benutzerdefinierten Rolle zusätzliche Rechte

```
db.grantPrivilegesToRole(
  "collectionRole",
  [
    {
      resource: { db: "sample-database-1", collection: "col1" },
      actions: ["find", "update", "insert", "remove"]
    }
  ]
)
```

Beispiel 11: Privilegien aus einer benutzerdefinierten Rolle entfernen

```
db.revokePrivilegesFromRole(
  "collectionRole",
  [
    {
      resource: { db: "sample-database-1", collection: "col2" },
      actions: ["find", "update", "insert", "remove"]
```

```
    }
  ]
)
```

Beispiel 12: Eine bestehende benutzerdefinierte Rolle aktualisieren

```
db.updateRole(
  "collectionRole",
  {
    privileges: [
    {
        resource: {db: "sample-database-3", collection: "sample-collection-3"},
 actions: ["find", "update", "insert", "remove"]
    }],
    roles: []
  }
)
```

Funktionale Unterschiede

In Amazon DocumentDB werden Benutzer- und Rollendefinitionen in der `admin` Datenbank gespeichert und Benutzer werden anhand der `admin` Datenbank authentifiziert. Diese Funktionalität unterscheidet sich von der MongoDB Community Edition, ist aber konsistent mit MongoDB Atlas.

Amazon DocumentDB unterstützt auch Change Streams, die eine zeitlich geordnete Abfolge von Änderungsereignissen bereitstellen, die in den Sammlungen Ihres Clusters auftreten. Die `listChangeStreams` Aktion wird auf Cluster-Ebene (d. h. auf alle Datenbanken) angewendet, und die `modifyChangeStreams` Aktion kann auf Datenbank- und Cluster-Ebene angewendet werden.

Einschränkungen

Die folgende Tabelle enthält die Grenzwerte für die Role-Based Zugriffskontrolle in Amazon DocumentDB.

Description	Limit
Anzahl Benutzer pro Cluster	1000
Anzahl der Rollen, die einem Benutzer zugeordnet sind	1000

Description	Limit
Anzahl der benutzerdefinierten Rollen	100
Anzahl der Ressourcen, die einem Privileg zugeordnet sind	100

Datenbankzugriff mit Role-Based Access Control

Mit der rollenbasierten Zugriffssteuerung können Sie einen Benutzer erstellen und ihm eine oder mehrere Rollen erteilen, um zu bestimmen, welche Operationen der Benutzer in einer Datenbank oder einem Cluster ausführen kann.

Im Folgenden finden Sie eine Liste der integrierten Rollen, die derzeit in Amazon DocumentDB unterstützt werden.

> **Note**
>
> In Amazon DocumentDB 4.0 und 5.0 können die `ListDatabase` Befehle `ListCollection` und optional die `authorizedDatabases` Parameter `authorizedCollections` und verwenden, um die Sammlungen und Datenbanken aufzulisten, auf die der Benutzer zugreifen darf, wobei die entsprechenden `listDatabase` Rollen `listCollections` und erforderlich sind. Außerdem haben Benutzer jetzt die Möglichkeit, ihre eigenen Cursor zu beenden, ohne die `KillCursor` Rolle zu benötigen.

Database user

Rollenname	Description	Aktionen

`read`	Gewährt einem Benutzer Lesezugriff auf die angegebene Datenbank.	`changeStreams`
		`collStats`
		`dbStats`
		`find`
		`killCursors`

Rollenname	Description	Aktionen
		`listIndexes` `listCollections`

`readWrite`	Gewährt dem Benutzer Lese- und Schreibzugriff auf die angegebene Datenbank.	Alle Aktionen von `read`-Berechtigungen.
		`createCollection`
		`dropCollection`
		`createIndex`
		`dropIndex`
		`insert`
		`killCursors`
		`listIndexes`
		`listCollections`
		`remove`
		`update`

Cluster user

Rollenname	Description	Aktionen
lesen AnyDatabase	Gewährt einem Benutzer Lesezugriff auf alle Datenbanken im Cluster.	Alle Aktionen von `read`-Berechtigungen.

Rollenname	Description	Aktionen
		`listChangeStreams` `listDatabases`

`readWriteAnyDatabase`	Gewährt einem Benutzer Lese- und Schreibzugriff auf alle Datenbanken im Cluster.	Alle Aktionen von `readWrite` - Berechtigungen.
		`listChangeStreams`
		`listDatabases`

`userAdminAnyDatabase`	Gewährt einem Benutzer die Möglichkeit, die Rollen oder Berechtigungen, die ein Benutzer hat, der angegebenen Datenbank zuzuweisen und zu ändern.	`changeCustomData`
		`changePassword`
		`createUser`
		`dropRole`
		`dropUser`
		`grantRole`
		`listDatabases`
		`revokeRole`
		`viewRole`
		`viewUser`

Rollenname	Description	Aktionen

`dbAdminAnyDatabase`	Gewährt einem Benutzer die Möglichkeit, Datenbankverwaltungsrollen für jede angegebene Datenbank auszuführen.	Alle Aktionen von `dbAdmin`-Berechtigungen.
		`dropCollection`
		`listDatabases`
		`listChangeStreams`
		`modifyChangeStreams`

Superuser

Rollenname	Description	Aktionen
`root`	Gewährt einem Benutzer Zugriff auf die Ressourcen und Operationen aller folgenden Rollen zusammen: `readWriteAnyDatabase` , `dbAdminAnyDatabase` , `userAdminAnyDatabase` , `clusterAdmin` , `restore` und `backup`.	Alle Aktionen von `readWriteAnyDatabase` , `dbAdminAnyDatabase` , `userAdminAnyDatabase` , `clusterAdmin` , `restore` und `backup`.

Database administrator

Rollenname	Description	Aktionen

`dbAdmin`	Gewährt einem Benutzer die Möglichkeit, administrative Aufgaben für die angegebene Datenbank auszuführen.	`bypassDocumentValidation`
		`collMod`
		`collStats`
		`createCollection`
		`createIndex`
		`dropCollection`
		`dropDatabase`
		`dropIndex`
		`dbStats`
		`find`
		`killCursors`
		`listIndexes`
		`listCollections`
		`modifyChangeStreams`

`dbOwner`	Gewährt einem Benutzer die Möglichkeit, alle administrativen Aufgaben für die angegebenen Datenbank auszuführen, indem die Rollen	Alle Aktionen von `dbAdmin` und `readWrite` .

Rollenname	Description	Aktionen
	dbAdmin und readWrite kombiniert werden.	

Cluster administrator

Name der Rolle	Description	Aktionen
clusterAdmin	Gewährt einem Benutzer den größtmöglichen Zugriff auf die Cluster-Verwaltung durch Kombination der Rollen clusterManager , clusterMonitor und hostManager .	Alle Aktionen von clusterManager , clusterMonitor und hostManager . listChangeStreams dropDatabase modifyChangeStreams
clusterManager	Gewährt einem Benutzer die Möglichkeit, Verwaltungs- und Überwachungsaktionen für den angegebenen Cluster durchzuführen.	listChangeStreams listSessions modifyChangeStreams replSetGetConfig
clusterMonitor	Gewährt einem Benutzer die Möglichkeit, schreibgeschützten Zugriff auf Überwachungstools zu haben.	collStats dbStats find getParameter

Name der Rolle	Description	Aktionen

		`hostInfo`
		`indexStats`
		`killCursors`
		`listChangeStreams`
		`listCollections`
		`listDatabases`
		`listIndexes`
		`listSessions`
		`replSetGetConfig`
		`serverStatus`
		`top`

`hostManager`	Gewährt einem Benutzer die Möglichkeit, Server zu überwachen und zu verwalten.	`auditConfigure`
		`killCursors`
		`killAnyCursor`
		`killAnySession`
		`killop`

Backup administrator

Rollenname	Description	Aktionen

`backup`	Gewährt einem Benutzer den erforderlichen Zugriff zum Sichern von Daten.	`getParameter`
		`insert`
		`find`
		`listChangeStreams`
		`listCollections`
		`listDatabases`
		`listIndexes`
		`update`

`restore`	Gewährt einem Benutzer den erforderlichen Zugriff zum Wiederherstellen von Daten.	`bypassDocumentValidation`
		`changeCustomData`
		`changePassword`
		`collMod`
		`createCollection`
		`createIndex`
		`createUser`
		`dropCollection`

Rollenname	Description	Aktionen

		`dropRole`
		`dropUser`
		`getParameter`
		`grantRole`
		`find`
		`insert`
		`listCollections`
		`modifyChangeStreams`
		`revokeRole`
		`remove`
		`viewRole`
		`viewUser`
		`update`

Aktualisierung Ihrer Amazon DocumentDB-TLS-Zertifikate

Topics

- Aktualisierung Ihrer Anwendung und Ihres Amazon DocumentDB-Clusters
- Automatische Rotation von Serverzertifikaten
- Fehlerbehebung
- Häufig gestellte Fragen

Das Zertifikat der Zertifizierungsstelle (CA) für Amazon DocumentDB-Cluster wurde im August 2024 aktualisiert. Wenn Sie Amazon DocumentDB-Cluster mit aktivierter Transport Layer Security (TLS) verwenden (Standardeinstellung) und Sie Ihre Client-Anwendungs- und Serverzertifikate nicht rotiert haben, sind die folgenden Schritte erforderlich, um Verbindungsprobleme zwischen Ihrer Anwendung und Ihren Amazon DocumentDB-Clustern zu beheben.

- Schritt 1: Herunterladen des neuen CA-Zertifikats und Aktualisieren Ihrer Anwendung
- Schritt 2: Aktualisieren des Serverzertifikats

Die CA- und Serverzertifikate wurden im Rahmen der bewährten Standardmethoden für Wartung und Sicherheit für Amazon DocumentDB aktualisiert. Client-Anwendungen müssen die neuen CA-Zertifikate zu ihren Trust Stores hinzufügen, und bestehende Amazon DocumentDB-Instances müssen vor diesem Ablaufdatum so aktualisiert werden, dass sie die neuen CA-Zertifikate verwenden.

Aktualisierung Ihrer Anwendung und Ihres Amazon DocumentDB-Clusters

Führen Sie die Schritte in diesem Abschnitt aus, um das CA-Zertifikatspaket Ihrer Anwendung (Schritt 1) und die Serverzertifikate Ihres Clusters (Schritt 2) zu aktualisieren. Bevor Sie die Änderungen auf Ihre Produktionsumgebungen anwenden, empfehlen wir dringend, diese Schritte in einer Entwicklungs- oder Stagingumgebung zu testen.

> **Note**
>
> Sie müssen jeweils die Schritte 1 und 2 ausführen, AWS-Region in denen Sie Amazon DocumentDB-Cluster haben.

Schritt 1: Herunterladen des neuen CA-Zertifikats und Aktualisieren Ihrer Anwendung

Laden Sie das neue CA-Zertifikat herunter und aktualisieren Sie Ihre Anwendung, um das neue CA-Zertifikat zum Erstellen von TLS-Verbindungen zu Amazon DocumentDB zu verwenden. Laden Sie das neue CA-Zertifikatpaket von https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem herunter. Diese Operation lädt die Datei mit dem Namen `global-bundle.pem` herunter.

> **Note**
>
> Wenn Sie auf den Keystore zugreifen, der sowohl das alte CA-Zertifikat (`rds-ca-2019-root.pem`) als auch die neuen CA-Zertifikate (`rds-ca-rsa2048-g1`,,) enthält, stellen Sie sicher`rds-ca-rsa4096-g1`, `rds-ca-ecc384-g1` dass der Keystore diese Option auswählt. `global-bundle`

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

Aktualisieren Sie als Nächstes Ihre Anwendungen, um das neue Zertifikatpaket zu verwenden. Das neue CA-Paket enthält sowohl das alte CA-Zertifikat (rds-ca-2019) als auch die neuen CA-Zertifikate (rds-ca-rsa2048-g1, rds-ca-rsa4096-g1, rds-ca-ecc384-g1). Wenn beide CA-Zertifikate im neuen CA-Paket enthalten sind, können Sie Ihre Anwendung und Ihren Cluster in zwei Schritten aktualisieren.

Für Java-Anwendungen müssen Sie einen neuen Trust Store mit dem neuen CA-Zertifikat erstellen. Anweisungen finden Sie auf der Registerkarte Java im Verbindung bei aktiviertem TLS herstellen Thema.

Informationen dazu, wie Sie überprüfen, ob Ihre Anwendung das neueste CA-Zertifikatspaket verwendet, finden Sie unter Wie kann ich sicher sein, dass ich das neueste Zertifizierungsstellenpaket verwende?. Wenn Sie das neueste CA-Zertifikatspaket in Ihrer Anwendung bereits verwenden, können Sie mit Schritt 2 fortfahren.

Beispiele für die Verwendung eines Zertifizierungsstellenpakets mit Ihrer Anwendung finden Sie unter Verschlüsseln von Daten während der Übertragung und Verbindung bei aktiviertem TLS herstellen.

> **Note**
>
> Zurzeit akzeptiert der MongoDB Go Driver 1.2.1 nur ein einzelnes CA-Serverzertifikat in `sslcertificateauthorityfile`. Informationen Verbindung bei aktiviertem TLS herstellen zur Verbindung mit Amazon DocumentDB mithilfe von Go, wenn TLS aktiviert ist, finden Sie unter.

Schritt 2: Aktualisieren des Serverzertifikats

Nachdem die Anwendung für die Verwendung des neuen CA-Bundles aktualisiert wurde, besteht der nächste Schritt darin, das Serverzertifikat zu aktualisieren, indem jede Instance in einem Amazon

DocumentDB-Cluster geändert wird. Informationen zum Ändern von Instances zur Verwendung des neuen Serverzertifikats finden Sie in den folgenden Anweisungen.

Amazon DocumentDB stellt die folgenden CAs zur Verfügung, um das DB-Serverzertifikat für eine DB-Instance zu signieren:

- rds-ca-ecc384-g1 — Verwendet eine Zertifizierungsstelle mit dem ECC 384-Algorithmus für private Schlüssel und dem SHA384-Signaturalgorithmus. Diese CA unterstützt die automatische Rotation von Serverzertifikaten. Dies wird nur in Amazon DocumentDB 4.0 und 5.0 unterstützt.
- rds-ca-rsa2048-g1 — Verwendet in den meisten Fällen eine Zertifizierungsstelle mit RSA 2048-Algorithmus für private Schlüssel und SHA256-Signaturalgorithmus. AWS-Regionen Diese CA unterstützt die automatische Rotation von Serverzertifikaten.
- rds-ca-rsa4096-g1 — Verwendet eine Zertifizierungsstelle mit dem RSA 4096-Algorithmus für private Schlüssel und dem SHA384-Signaturalgorithmus. Diese CA unterstützt die automatische Rotation von Serverzertifikaten.

> **Note**
>
> Wenn Sie den verwenden, können Sie die Gültigkeiten der im AWS CLI vorherigen Abschnitt aufgeführten Zertifizierungsstellen mithilfe von describe-certificates überprüfen. https://docs.aws.amazon.com/cli/latest/reference/docdb/describe-certificates.html

Diese CA-Zertifikate sind im regionalen und globalen Zertifikatpakets enthalten. Wenn Sie die rds-ca-rsa2048-g1-, rds-ca-rsa4096-g1- oder rds-ca-ecc384-g1 CA mit einer Datenbank verwenden, verwaltet Amazon DocumentDB das DB-Serverzertifikat in der Datenbank. Amazon DocumentDB rotiert das DB-Serverzertifikat automatisch, bevor es abläuft.

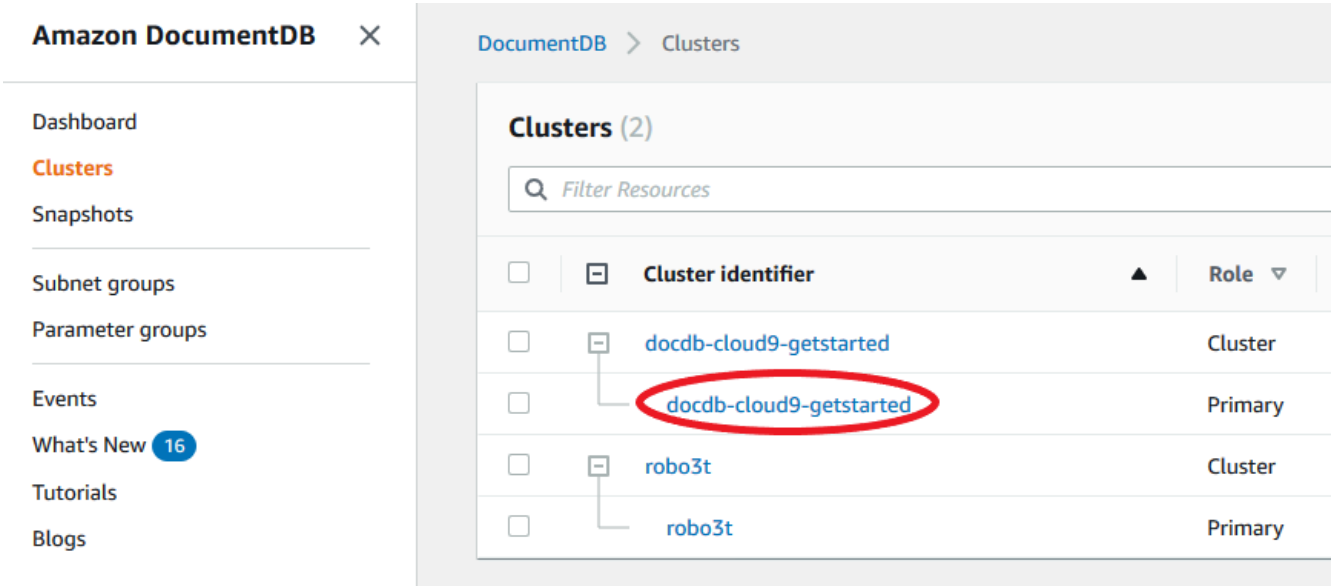
> **Note**
>
> Amazon DocumentDB erfordert keinen Neustart für die Zertifikatsrotation, wenn Ihr Cluster auf den folgenden Engine-Patch-Versionen läuft:
>
> - Amazon DocumentDB 3.6:1.0.208662 oder höher
> - Amazon DocumentDB 4.0:2.0.10179 oder höher
> - Amazon DocumentDB 5.0:3.0.4780 oder höher

Sie können die aktuelle Patch-Version der Amazon DocumentDB-Engine ermitteln, indem Sie den folgenden Befehl ausführen:. `db.runCommand({getEngineVersion: 1})` Stellen Sie vor dem Aktualisieren des Serverzertifikats sicher, dass Sie den Vorgang abgeschlossen Schritt 1 haben.

Using the AWS-Managementkonsole

Führen Sie die folgenden Schritte aus, um das alte Serverzertifikat für Ihre vorhandenen Amazon DocumentDB-Instances zu identifizieren und zu rotieren. Verwenden Sie dazu die AWS-Managementkonsole.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Wählen Sie in der Liste der Regionen in der oberen rechten Ecke des Bildschirms die Regionen aus, AWS-Region in denen sich Ihre Cluster befinden.
3. Wählen Sie im Navigationsbereich auf der linken Seite der Konsole die Option Cluster aus.
4. Möglicherweise müssen Sie ermitteln, welche Instanzen sich noch auf dem alten Serverzertifikat befinden (). `rds-ca-2019` Sie können dies in der Spalte Zertifizierungsstelle tun, die sich ganz rechts in der Cluster-Tabelle befindet.
5. In der Cluster-Tabelle sehen Sie die Spalte Cluster-ID ganz links. Ihre Instances sind unter Clustern aufgeführt, ähnlich wie im folgenden Screenshot.



6. Markieren Sie das Kästchen links neben der Instanz, an der Sie interessiert sind.
7. Wählen Sie Aktionen und dann Ändern.
8. Wählen Sie unter Certificate authority (Zertifizierungsstelle) das neue Serverzertifikat (d. h. `rds-ca-rsa2048-g1`) für diese Instance aus.
9. Sie können eine Zusammenfassung der Änderungen auf der nächsten Seite sehen. Beachten Sie, dass es eine zusätzliche Warnung gibt, die Sie daran erinnert, dass Ihre Anwendung das neueste Zertifizierungsstellenpaket verwendet, bevor Sie die Instance ändern, um eine Unterbrechung der Konnektivität zu vermeiden.
10. Sie können die Änderung während Ihres nächsten Wartungsfensters oder sofort anwenden. Wenn Sie beabsichtigen, das Serverzertifikat sofort zu ändern, verwenden Sie die Option Apply Immediately (Sofort anwenden) .
11. Wählen Sie Modify instance (Instance ändern), um die Aktualisierung abzuschließen.

Using the AWS CLI

Führen Sie die folgenden Schritte aus, um das alte Serverzertifikat für Ihre vorhandenen Amazon DocumentDB-Instances mithilfe von zu identifizieren und zu rotieren. AWS CLI

1. Um die Instances sofort zu ändern, führen Sie für jede Instance im Cluster den folgenden Befehl aus.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier>
 --ca-certificate-identifier rds-ca-rsa2048-g1 --apply-immediately
```

2. Um die Instances in Ihren Clustern so zu ändern, dass sie während des nächsten Wartungsfensters Ihres Clusters das neue CA-Zertifikat verwenden, führen Sie für jede Instance im Cluster den folgenden Befehl aus.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier>
 --ca-certificate-identifier rds-ca-rsa2048-g1 --no-apply-immediately
```

Automatische Rotation von Serverzertifikaten

Amazon DocumentDB unterstützt die automatische Rotation von Serverzertifikaten. Das Serverzertifikat ist das Leaf-Zertifikat, das für jede Cluster-Instance ausgestellt wird. Im Gegensatz zu den Root-CA-Zertifikaten haben die Serverzertifikate eine kurze Gültigkeitsdauer (12 Monate),

und Amazon DocumentDB kümmert sich automatisch um ihre Rotation, ohne dass Sie etwas unternehmen müssen. Amazon DocumentDB verwendet dieselbe Root-CA für diese automatische Rotation, sodass Sie kein neues CA-Paket herunterladen müssen.

> **Important**
>
> Wenn Sie eine Verbindung zu Ihrem Amazon DocumentDB-Cluster herstellen, vertrauen Sie dem Root-CA-Bundle, anstatt jedem Serverzertifikat direkt zu vertrauen. Dadurch werden Verbindungsfehler vermieden, nachdem das Serverzertifikat rotiert wurde. Siehe Verbindung bei aktiviertem TLS herstellen.

Amazon DocumentDB versucht, Ihr Serverzertifikat in Ihrem bevorzugten Wartungsfenster zu rotieren, sobald die Halbwertszeit des Serverzertifikats abgelaufen ist. Das neue Serverzertifikat ist 12 Monate gültig.

Verwenden Sie den `describe-db-engine-versions` Befehl und überprüfen Sie das `SupportsCertificateRotationWithoutRestart` Flag, um festzustellen, ob die Engine-Version das Rotieren des Zertifikats ohne Neustart unterstützt.

> **Note**
>
> Amazon DocumentDB unterstützt die Rotation von Serverzertifikaten ohne Neustarts, wenn Ihr Cluster auf den folgenden Engine-Patch-Versionen läuft:
>
> - Amazon DocumentDB 3.6:1.0.208662 oder höher
> - Amazon DocumentDB 4.0:2.0.10179 oder höher
> - Amazon DocumentDB 5.0:3.0.4780 oder höher
>
> Sie können die aktuelle Patch-Version der Amazon DocumentDB-Engine ermitteln, indem Sie diesen Befehl ausführen:. `db.runCommand({getEngineVersion: 1})`
> Wenn Sie eine ältere Engine-Patch-Version verwenden, rotiert Amazon DocumentDB das Serverzertifikat und plant in Ihrem bevorzugten Wartungsfenster einen Datenbank-Neustart.

Fehlerbehebung

Wenn Sie im Rahmen der Zertifikatsrotation Probleme haben, eine Verbindung zu Ihrem Cluster herzustellen, versuchen Sie Folgendes:

- Stellen Sie sicher, dass Ihre Clients das neueste Zertifikatpaket verwenden. Siehe Wie kann ich sicher sein, dass ich das neueste Zertifizierungsstellenpaket verwende?.
- Stellen Sie sicher, dass Ihre Instances das neueste Zertifikat verwenden. Siehe Woher weiß ich, welche meiner Amazon DocumentDB-Instances das old/new Serverzertifikat verwenden?.
- Stellen Sie sicher, dass die neueste Zertifizierungsstelle von Ihrer Anwendung verwendet wird. Einige Treiber, wie Java und Go, benötigen zusätzlichen Code, um mehrere Zertifikate aus einem Zertifikatpaket in den Vertrauensspeicher zu importieren. Weitere Informationen zur Verbindung mit Amazon DocumentDB mit TLS finden Sie unterProgrammgesteuertes Herstellen einer Verbindung zu Amazon DocumentDB.
- Wenden Sie sich an den Support. Wenn Sie Fragen oder Probleme haben, wenden Sie sich an Support.

Häufig gestellte Fragen

Im Folgenden finden Sie Antworten auf einige häufig gestellte Fragen zu TLS-Zertifikaten.

Was passiert, wenn ich Fragen oder Probleme habe?

Wenn Sie Fragen oder Probleme haben, wenden Sie sich an Support.

Woher weiß ich, ob ich TLS verwende, um eine Verbindung zu meinem Amazon DocumentDB-Cluster herzustellen?

Sie können bestimmen, ob Ihr Cluster TLS verwendet, indem Sie den `tls`-Parameter für die Clusterparametergruppe Ihres Clusters untersuchen. Wenn der `tls`-Parameter auf `enabled` festgelegt ist, verwenden Sie das TLS-Zertifikat, um eine Verbindung mit dem Cluster herzustellen. Weitere Informationen finden Sie unter Verwaltung von Amazon DocumentDB-Cluster-Parametergruppen.

Warum aktualisieren Sie die Zertifizierungsstellen- und Serverzertifikate?

Die Amazon DocumentDB-Zertifizierungsstelle und die Serverzertifikate werden im Rahmen der bewährten Standardmethoden für Wartung und Sicherheit für Amazon DocumentDB aktualisiert.

Was passiert, wenn ich bis zum Ablaufdatum keine Maßnahmen ergreife?

Wenn Sie TLS verwenden, um mit einem abgelaufenen CA-Zertifikat eine Verbindung zu Ihrem Amazon DocumentDB-Cluster herzustellen, können Ihre Anwendungen, die eine Verbindung über TLS herstellen, nicht mehr mit dem Amazon DocumentDB-Cluster kommunizieren.

Amazon DocumentDB rotiert Ihre Datenbankzertifikate vor Ablauf nicht automatisch. Sie müssen Ihre Anwendungen und Cluster aktualisieren, um die neuen CA-Zertifikate vor oder nach dem Ablaufdatum verwenden zu können.

Woher weiß ich, welche meiner Amazon DocumentDB-Instances das old/new Serverzertifikat verwenden?

Um die Amazon DocumentDB-Instances zu identifizieren, die noch das alte Serverzertifikat verwenden, können Sie entweder die Amazon DocumentDB AWS-Managementkonsole oder die verwenden. AWS CLI

Verwendung der AWS-Managementkonsole

So identifizieren Sie die Instances in Ihren Clustern, die das ältere Zertifikat verwenden

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Wählen Sie in der Liste der Regionen in der oberen rechten Ecke des Bildschirms die Region aus, AWS-Region in der sich Ihre Instances befinden.
3. Wählen Sie im Navigationsbereich auf der linken Seite der Konsole die Option Cluster aus.
4. In der Spalte Zertifizierungsstelle (ganz rechts in der Tabelle) wird angezeigt, welche Instances sich noch auf dem alten Serverzertifikat (`rds-ca-2019`) und dem neuen Serverzertifikat () befinden. `rds-ca-rsa2048-g1`

Verwendung der AWS CLI

Um die Instances in Ihren Clustern zu identifizieren, die das ältere Serverzertifikat verwenden, verwenden Sie den Befehl `describe-db-clusters` mit Folgendem:

```
aws docdb describe-db-instances \
    --filters Name=engine,Values=docdb \
```

```
    --query 'DBInstances[*].
{CertificateVersion:CACertificateIdentifier,InstanceID:DBInstanceIdentifier}'
```

Wie ändere ich einzelne Instances in meinem Amazon DocumentDB-Cluster, um das Serverzertifikat zu aktualisieren?

Aktualisieren Sie Serverzertifikate für alle Instances in einem bestimmten Cluster gleichzeitig. Um die Instances im Cluster zu ändern, können Sie die Konsole oder die AWS CLI verwenden.

> **Note**
>
> Sie müssen Schritt 1 abschließen, bevor Sie das Serverzertifikat aktualisieren.

Verwendung der AWS-Managementkonsole

1. Melden Sie sich bei und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb. AWS-Managementkonsole
2. Wählen Sie in der Liste der Regionen in der oberen rechten Ecke des Bildschirms die Regionen aus, AWS-Region in denen sich Ihre Cluster befinden.
3. Wählen Sie im Navigationsbereich auf der linken Seite der Konsole die Option Cluster aus.
4. In der Spalte Zertifizierungsstelle (ganz rechts in der Tabelle) wird angezeigt, welche Instances sich noch auf dem alten Serverzertifikat () befinden. `rds-ca-2019`
5. Wählen Sie in der Cluster-Tabelle unter Cluster-ID eine Instanz aus, die Sie ändern möchten.
6. Wählen Sie Aktionen und dann Ändern.
7. Wählen Sie unter Certificate authority (Zertifizierungsstelle) das neue Serverzertifikat (d. h. `rds-ca-rsa2048-g1`) für diese Instance aus.
8. Sie können eine Zusammenfassung der Änderungen auf der nächsten Seite sehen. Beachten Sie, dass es eine zusätzliche Warnung gibt, die Sie daran erinnert, dass Ihre Anwendung das neueste Zertifizierungsstellenpaket verwendet, bevor Sie die Instance ändern, um eine Unterbrechung der Konnektivität zu vermeiden.
9. Sie können die Änderung während Ihres nächsten Wartungsfensters oder sofort anwenden.
10. Wählen Sie Modify instance (Instance ändern), um die Aktualisierung abzuschließen.

Verwendung der AWS CLI

Führen Sie die folgenden Schritte aus, um das alte Serverzertifikat für Ihre vorhandenen Amazon DocumentDB-Instances zu identifizieren und zu rotieren. Verwenden Sie dazu die AWS CLI.

1. Um die Instances sofort zu ändern, führen Sie für jede Instance im Cluster den folgenden Befehl aus.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --
ca-certificate-identifier rds-ca-rsa2048-g1 --apply-immediately
```

2. Um die Instances in Ihren Clustern so zu ändern, dass sie während des nächsten Wartungsfensters Ihres Clusters das neue CA-Zertifikat verwenden, führen Sie für jede Instance im Cluster den folgenden Befehl aus.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --
ca-certificate-identifier rds-ca-rsa2048-g1 --no-apply-immediately
```

Was passiert, wenn ich einem vorhandenen Cluster eine neue Instance hinzufüge?

Alle neu erstellten Instances verwenden das alte Serverzertifikat und erfordern TLS-Verbindungen über das alte CA-Zertifikat. Alle neuen Amazon DocumentDB-Instances, die nach dem 25. Januar 2024 erstellt werden, verwenden standardmäßig das neue Zertifikat rds-ca-rsa2048-g1.

Was passiert, wenn ein Instance-Ersatz oder ein Failover auf meinem Cluster vorhanden ist?

Wenn ein Instance-Ersatz in Ihrem Cluster vorhanden ist, verwendet die neu erstellte Instance weiterhin dasselbe Serverzertifikat wie die Instance davor. Es wird empfohlen, Serverzertifikate für alle Instances gleichzeitig zu aktualisieren. Wenn ein Failover im Cluster auftritt, wird das Serverzertifikat auf dem neuen Primärserver verwendet.

Wenn ich keine TLS für die Verbindung mit meinem Cluster verwende, muss ich trotzdem jede meiner Instances aktualisieren?

Wir empfehlen dringend, TLS zu aktivieren. Für den Fall, dass Sie TLS nicht aktivieren, empfehlen wir dennoch, die Zertifikate auf Ihren Amazon DocumentDB-Instances zu rotieren, falls Sie in Zukunft TLS für die Verbindung zu Ihren Clustern verwenden möchten. Wenn Sie niemals vorhaben, TLS

für die Verbindung zu Ihren Amazon DocumentDB-Clustern zu verwenden, sind keine Maßnahmen erforderlich.

Was soll ich tun, wenn ich derzeit nicht TLS für die Verbindung mit meinem Cluster verwende, dies zukünftig aber vorhabe?

Wenn Sie vor Januar 2024 einen Cluster erstellt haben, folgen Sie Schritt 1 und Schritt 2 im vorherigen Abschnitt, um sicherzustellen, dass Ihre Anwendung das aktualisierte CA-Paket verwendet und dass jede Amazon DocumentDB-Instance das neueste Serverzertifikat verwendet. Wenn Sie nach dem 25. Januar 2024 einen Cluster erstellen, verfügt Ihr Cluster bereits über das neueste Serverzertifikat (rds-ca-rsa2048-g1). Informationen zum Überprüfen, ob Ihre Anwendung das neueste CA-Paket verwendet, finden Sie unter Wenn ich keine TLS für die Verbindung mit meinem Cluster verwende, muss ich trotzdem jede meiner Instances aktualisieren?.

Kann die Frist über August 2024 hinaus verlängert werden?

Wenn Ihre Bewerbungen über TLS verbunden sind, kann die Frist nicht verlängert werden.

Wie kann ich sicher sein, dass ich das neueste Zertifizierungsstellenpaket verwende?

Verwenden Sie den folgenden Befehl, um zu überprüfen, ob Sie das neueste Paket haben. Um diesen Befehl auszuführen, müssen Sie Java installiert haben und die Java-Tools müssen sich in der PATH-Variablen Ihrer Shell befinden. Weitere Informationen finden Sie unter Java verwenden

macOS und Amazon Linux

```
keytool -printcert -v -file global-bundle.pem
```

Windows

```
keytool -printcert -v -file global-bundle.p7b
```

Warum sehe ich „RDS“ im Namen des Zertifizierungsstellenpakets?

Für bestimmte Verwaltungsfunktionen, wie z. B. die Zertifikatsverwaltung, verwendet Amazon DocumentDB Betriebstechnologie, die gemeinsam mit Amazon Relational Database Service (Amazon RDS) genutzt wird.

Wann läuft das neue Zertifikat ab?

Das neue Serverzertifikat läuft (in der Regel) wie folgt ab:

- rds-ca-rsa2048-g1 — Läuft 2016 ab
- rds-ca-rsa4096-g1 — Läuft 2121 ab
- rds-ca-ecc384-g1 — Läuft 2121 ab

Welche Fehler werden mir angezeigt, wenn ich vor Ablauf des Zertifikats keine Maßnahmen ergreife?

Die Fehlermeldungen variieren je nach Treiber. Im Allgemeinen werden Fehler bei der Zertifikatsvalidierung angezeigt, die die Zeichenfolge „Zertifikat ist abgelaufen“ enthalten.

Kann ich nach der Anwendung des neuen Serverzertifikats wieder zum alten Zertifikat zurückkehren?

Wenn Sie eine Instanz auf das alte Serverzertifikat zurücksetzen müssen, tun Sie dies für alle Instanzen im Cluster. Sie können das Serverzertifikat für jede Instanz in einem Cluster zurücksetzen, indem Sie die AWS-Managementkonsole oder verwenden. AWS CLI

Verwendung der AWS-Managementkonsole

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Wählen Sie in der Liste der Regionen in der oberen rechten Ecke des Bildschirms die Regionen aus, AWS-Region in denen sich Ihre Cluster befinden.
3. Wählen Sie im Navigationsbereich auf der linken Seite der Konsole die Option Cluster aus.
4. Wählen Sie in der Cluster-Tabelle unter Cluster-ID eine Instance aus, die Sie ändern möchten. Wählen Sie Actions (Aktionen) und dann Modify (Ändern) aus.
5. Unter Certificate authority (Zertifizierungsstelle) können Sie das alte Serverzertifikat (`rds-ca-2019`) auswählen.
6. Wählen Sie Continue (Weiter) aus, um eine Übersicht Ihrer Änderungen anzuzeigen.
7. Auf dieser resultierenden Seite können Sie festlegen, dass Ihre Änderungen im nächsten Wartungsfenster oder sofort angewendet werden sollen. Wählen Sie die gewünschte Option und anschließend Modify instance (Instance ändern) aus.

> **Note**
>
> Wenn Sie Ihre Änderungen sofort anwenden möchten, werden alle Änderungen in der Warteschlange für ausstehende Änderungen ebenfalls angewendet. Wenn eine der ausstehenden Änderungen eine Ausfallszeit erfordert, kann die Auswahl dieser Option einen unerwarteten Ausfall verursachen.

Verwendung der AWS CLI

```
aws docdb modify-db-instance --db-instance-identifier <db_instance_name> ca-
certificate-identifier rds-ca-2019 <--apply-immediately | --no-apply-immediately>
```

Wenn Sie `--no-apply-immediately` wählen, werden die Änderungen während des nächsten Wartungsfensters des Clusters angewendet.

Wird bei einer Wiederherstellung, die ich aus einem Snapshot oder zeitpunktbezogen ausführe, das neue Serverzertifikat verwendet?

Wenn Sie nach August 2024 einen Snapshot wiederherstellen oder eine Point-in-Time-Wiederherstellung durchführen, verwendet der neu erstellte Cluster das neue CA-Zertifikat.

Was ist, wenn ich Probleme habe, von einem beliebigen Mac OS aus eine direkte Verbindung zu meinem Amazon DocumentDB-Cluster herzustellen?

Mac OS hat die Anforderungen für vertrauenswürdige Zertifikate aktualisiert. Vertrauenswürdige Zertifikate müssen jetzt 397 Tage oder weniger gültig sein (siehehttps://support.apple.com/en-us/HT211025).

> **Note**
>
> Diese Einschränkung wird in neueren Versionen von Mac OS eingehalten.

Amazon DocumentDB-Instance-Zertifikate sind über vier Jahre gültig, länger als das Maximum für Mac OS. Um von einem Computer mit Mac OS aus eine direkte Verbindung zu einem Amazon DocumentDB-Cluster herzustellen, müssen Sie beim Erstellen der TLS-Verbindung ungültige Zertifikate zulassen. In diesem Fall bedeuten ungültige Zertifikate, dass die Gültigkeitsdauer länger

als 397 Tage ist. Sie sollten die Risiken verstehen, bevor Sie ungültige Zertifikate zulassen, wenn Sie eine Verbindung zu Ihrem Amazon DocumentDB-Cluster herstellen.

Verwenden Sie den Parameter, um von Mac OS aus eine Verbindung zu einem Amazon DocumentDB-Cluster herzustellen. AWS CLI`tlsAllowInvalidCertificates`

```
mongo --tls --host <hostname> --username <username> --password <password> --port 27017
 --tlsAllowInvalidCertificates
```

Überprüfung der Konformität in Amazon DocumentDB

Die Sicherheit und Konformität von Amazon DocumentDB werden von externen Auditoren im Rahmen mehrerer AWS Compliance-Programme bewertet, darunter die folgenden:

- System and Organization Controls (SOC) 1, 2 und 3. Weitere Informationen finden Sie unter SOC.
- Bundesprogramm für Risiko- und Autorisierungsmanagement (FedRAMP). Weitere Informationen finden Sie unterAWS -Services im Rahmen des Compliance-Programms.
- Payment Card Industry Data Security Standard (PCI DSS). Weitere Informationen finden Sie unter PCI DSS.
- ISO 9001, 27001, 27017 und 27018. Weitere Informationen finden Sie unter ISO-zertifiziert.
- Health Insurance Portability and Accountability Act Business Associate Agreement (HIPAA BAA). Weitere Informationen finden Sie unter HIPAA-Compliance.

AWS bietet eine häufig aktualisierte Liste der AWS Dienste im Rahmen bestimmter Compliance-Programme unter AWS Services in Scope by Compliance Program.

Third-party Auditberichte stehen Ihnen mit Hilfe von zum Download zur Verfügung AWS Artifact. Weitere Informationen finden Sie unter Berichte in AWS Artifact https://docs.aws.amazon.com/artifact/latest/ug/downloading-documents.html herunterladen.

Weitere Informationen zu AWS Compliance-Programmen finden Sie unter AWS Compliance-Programme.

Ihre Compliance-Verantwortung bei der Verwendung von Amazon DocumentDB hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihrer Organisation sowie den geltenden Gesetzen und Vorschriften ab. Wenn Ihre Nutzung von Amazon DocumentDB von der Einhaltung von Standards wie HIPAA oder PCI abhängig ist, finden Sie hier hilfreiche Ressourcen AWS :

- AWS Compliance-Ressourcen — Eine Sammlung von Arbeitsmappen und Leitfäden, die für Ihre Branche und Ihren Standort gelten könnten.
- Schnellstartanleitungen zu Sicherheit und Compliance — Bereitstellungsleitfäden, in denen architektonische Überlegungen erörtert und Schritte für die Bereitstellung sicherheits- und Compliance-orientierter Basisumgebungen beschrieben werden. AWS
- AWS Config — Ein Service, der bewertet, wie gut Ihre Ressourcenkonfigurationen internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- AWS Security Hub — Ein umfassender Überblick über Ihren Sicherheitsstatus, der Ihnen hilft AWS , Ihre Einhaltung der branchenüblichen Sicherheitsstandards und Best Practices zu überprüfen.
- Whitepaper „Architecting for HIPAA Security and Compliance“ — Ein Whitepaper, das beschreibt, wie Unternehmen Anwendungen erstellen können. AWS HIPAA-compliant

Resilienz in Amazon DocumentDB

Die AWS globale Infrastruktur basiert auf AWS-Regionen Availability Zones. AWS-Regionen stellen mehrere physisch getrennte und isolierte Availability Zones bereit, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hochredundanten Netzwerken miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Availability Zones ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser hoch verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Ein Amazon DocumentDB-Cluster kann nur in einer Amazon VPC erstellt werden, die über mindestens zwei Subnetze in mindestens zwei Availability Zones verfügt. Durch die Verteilung Ihrer Cluster-Instances auf mindestens zwei Availability Zones trägt Amazon DocumentDB dazu bei, dass im unwahrscheinlichen Fall eines Ausfalls der Availability Zone Instances in Ihrem Cluster verfügbar sind. Das Cluster-Volume für Ihren Amazon DocumentDB-Cluster erstreckt sich immer über drei Availability Zones, um einen dauerhaften Speicher bereitzustellen, bei dem die Gefahr von Datenverlusten geringer ist.

Weitere Informationen zu AWS-Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Zusätzlich zur AWS globalen Infrastruktur bietet Amazon DocumentDB mehrere Funktionen, mit denen Sie Ihre Anforderungen an Datenstabilität und Datensicherung erfüllen können.

Fault-tolerant und selbstheilender Speicher

Jeder 10-GB-Teil Ihres Speichervolumens wird auf sechs Arten in drei Availability Zones repliziert. Amazon DocumentDB verwendet einen fehlertoleranten Speicher, der den Verlust von bis zu zwei Datenkopien transparent verarbeitet, ohne die Schreibverfügbarkeit der Datenbank zu beeinträchtigen, und bis zu drei Kopien, ohne die Leseverfügbarkeit zu beeinträchtigen. Der Amazon DocumentDB-Speicher repariert sich ebenfalls selbst. Datenblöcke und Festplatten werden kontinuierlich auf Fehler überprüft und automatisch ersetzt.

Manuelle Sicherungen und Wiederherstellung

Amazon DocumentDB bietet die Möglichkeit, vollständige Backups Ihres Clusters zur langfristigen Aufbewahrung und Wiederherstellung zu erstellen. Weitere Informationen finden Sie unter Sichern und Wiederherstellen in Amazon DocumentDB.

Point-in-time Wiederherstellung

Point-in-time Die Wiederherstellung schützt Ihre Amazon DocumentDB-Cluster vor versehentlichen Schreib- oder Löschvorgängen. Mit der zeitpunktbezogenen Wiederherstellung müssen Sie sich keine Gedanken über das Erstellen, Warten oder Planen von On-Demand-Backups machen. Weitere Informationen finden Sie unter Wiederherstellung auf einen bestimmten Zeitpunkt.

Infrastruktursicherheit in Amazon DocumentDB

Als verwalteter Service ist Amazon DocumentDB durch AWS globale Netzwerksicherheit geschützt. Informationen zu AWS Sicherheitsdiensten und zum AWS Schutz der Infrastruktur finden Sie unter AWS Cloud-Sicherheit. Informationen zum Entwerfen Ihrer AWS Umgebung anhand der Best Practices für die Infrastruktursicherheit finden Sie unter Infrastrukturschutz im Security Pillar AWS Well-Architected Framework.

Sie verwenden AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf Amazon DocumentDB zuzugreifen. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Verschlüsselungssammlungen mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Diffie-Hellman Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Sie können diese API-Vorgänge von einem beliebigen Netzwerkstandort aus aufrufen. Sie können Amazon DocumentDB-Richtlinien verwenden, um den Zugriff von bestimmten Amazon Virtual Private Cloud (Amazon VPC) -Endpunkten oder bestimmten VPCs aus zu kontrollieren. Dadurch wird der Netzwerkzugriff auf eine bestimmte Amazon DocumentDB-Ressource effektiv nur von der spezifischen VPC innerhalb des Netzwerks isoliert. AWS

> **Note**
>
> Amazon DocumentDB unterstützt keine ressourcenbasierten Zugriffsrichtlinien.

Amazon DocumentDB-API und Schnittstellen-VPC-Endpunkte (AWS PrivateLink)

> **Note**
>
> Elastische Amazon DocumentDB-Cluster unterstützen keine AWS PrivateLink VPC-Endpunkte.

Sie können eine private Verbindung zwischen Ihrer VPC und den Amazon DocumentDB-API-Endpunkten herstellen, indem Sie einen Schnittstellen-VPC-Endpunkt erstellen. Schnittstellen-Endpunkte werden unterstützt von. AWS PrivateLink

Auf Amazon DocumentDB-Instances basierende Cluster benötigen zwar keine VPC-Schnittstellen-Endpunktverbindung, AWS PrivateLink ermöglichen Ihnen jedoch den privaten Zugriff auf Amazon DocumentDB-API-Operationen ohne Internet-Gateway, NAT-Gerät, VPN-Verbindung oder Verbindung. Direct Connect Amazon DocumentDB-Instances in Ihrer VPC benötigen keine öffentlichen IP-Adressen, um mit Amazon DocumentDB-API-Endpunkten zu kommunizieren, um Datenbank-Instances und Datenbank-Cluster zu starten, zu ändern oder zu beenden. Ihre Amazon DocumentDB-Instances benötigen auch keine öffentlichen IP-Adressen, um die verfügbaren Amazon DocumentDB-API-Operationen nutzen zu können. Der Datenverkehr zwischen Ihrer VPC und Amazon DocumentDB verlässt das Amazon-Netzwerk nicht.

Jeder Schnittstellenendpunkt wird durch eine oder mehrere Elastic Network-Schnittstellen in Ihren Subnetzen dargestellt. Weitere Informationen finden Sie unter Elastic Network-Schnittstellen im Benutzerhandbuch für Amazon EC2.

Weitere Informationen zu VPC-Endpunkten finden Sie unter Zugreifen auf und AWS-Service mithilfe eines Schnittstellen-VPC-Endpunkts im Amazon Virtual Private Cloud () -Benutzerhandbuch.AWS PrivateLink Weitere Informationen zu Amazon DocumentDB-Vorgängen finden Sie in der Amazon DocumentDB-API-Referenz.

Topics

- Überlegungen zu VPC-Endpunkten
- Verfügbarkeit in Regionen
- Erstellen eines VPC-Schnittstellen-Endpunkts für die Amazon DocumentDB-API
- Erstellen einer VPC-Endpunktrichtlinie für die Amazon DocumentDB-API

Überlegungen zu VPC-Endpunkten

Bevor Sie einen Schnittstellen-VPC-Endpunkt für Amazon DocumentDB-API-Endpunkte einrichten, stellen Sie sicher, dass Sie die Voraussetzungen für Schnittstellenendpunkte im Amazon Virtual Private Cloud () -Benutzerhandbuch überprüfen.AWS PrivateLink

Alle Amazon DocumentDB-API-Operationen, die für die Verwaltung von Amazon DocumentDB-Ressourcen relevant sind, sind über Ihre VPC verfügbar. AWS PrivateLink

VPC-Endpunktrichtlinien werden für Amazon DocumentDB-API-Endpunkte unterstützt. Standardmäßig ist der vollständige Zugriff auf Amazon DocumentDB-API-Operationen über den Endpunkt zulässig. Weitere Informationen finden Sie unter Steuern des Zugriffs auf VPC-Endpunkte mithilfe von Endpunktrichtlinien im Amazon Virtual Private Cloud (AWS PrivateLink) -Benutzerhandbuch.

Verfügbarkeit in Regionen

Die Amazon DocumentDB-API unterstützt derzeit VPC-Endpunkte in den folgenden Bereichen: AWS-Regionen

- US East (Ohio)
- USA Ost (Nord-Virginia)
- USA West (Oregon)
- Africa (Cape Town)
- Asia Pacific (Hong Kong)
- Asien-Pazifik (Mumbai)

- Asien-Pazifik (Hyderabad)
- Asien-Pazifik (Osaka)
- Asien-Pazifik (Seoul)
- Asien-Pazifik (Singapur)
- Asien-Pazifik (Sydney)
- Asien-Pazifik (Tokio)
- Canada (Central)
- China (Peking)
- China (Ningxia)
- Europe (Frankfurt)
- Europa (Irland)
- Europa (London)
- Europa (Paris)
- Europa (Spain)
- Europa (Milan)
- Naher Osten (VAE)
- Südamerika (São Paulo)
- AWS GovCloud (US-East)
- AWS GovCloud (US-West)

Erstellen eines VPC-Schnittstellen-Endpunkts für die Amazon DocumentDB-API

Sie können einen VPC-Endpunkt für die Amazon DocumentDB-API erstellen, indem Sie entweder die Amazon VPC-Konsole oder die () verwenden. AWS Command Line Interface AWS CLI Weitere Informationen finden Sie im Amazon Virtual Private Cloud (AWS PrivateLink) -Benutzerhandbuch unter Zugreifen auf und AWS-Service mithilfe eines VPC-Schnittstellen-Endpunkts.

Erstellen Sie mithilfe des Servicenamens einen VPC-Endpunkt für die Amazon DocumentDB-API. `com.amazonaws.region.rds`

Außer AWS-Regionen in China können Sie, wenn Sie privates DNS für den Endpunkt aktivieren, API-Anfragen an Amazon DocumentDB stellen, wobei der VPC-Endpunkt seinen Standard-DNS-Namen

für den verwendet AWS-Regionen, zum Beispiel rds.us-east-1.amazonaws.com. Für China (Peking) und China (Ningxia) AWS-Regionen können Sie API-Anfragen mit dem VPC-Endpunkt über rds-api.cn-north-1.amazonaws.com .cn bzw. rds-api.cn-northwest-1.amazonaws.com .cn stellen.

Weitere Informationen finden Sie im Amazon Virtual Private Cloud () -Benutzerhandbuch unter Zugreifen auf und AWS-Service mithilfe eines Schnittstellen-VPC-Endpunkts.AWS PrivateLink

Erstellen einer VPC-Endpunktrichtlinie für die Amazon DocumentDB-API

Sie können Ihrem VPC-Endpunkt eine Endpunktrichtlinie hinzufügen, die den Zugriff auf die Amazon DocumentDB-API steuert. Die Richtlinie gibt die folgenden Informationen an:

- Prinzipal, der die Aktionen ausführen kann.
- Aktionen, die ausgeführt werden können
- Die Ressourcen, für die Aktionen ausgeführt werden können.

Weitere Informationen finden Sie unter Steuern des Zugriffs auf VPC-Endpunkte mithilfe von Endpunktrichtlinien im Amazon Virtual Private Cloud ()AWS PrivateLink-Benutzerhandbuch.

Beispiel: VPC-Endpunktrichtlinie für Amazon DocumentDB-API-Aktionen

Im Folgenden finden Sie ein Beispiel für eine Endpunktrichtlinie für die Amazon DocumentDB-API. Wenn diese Richtlinie an einen Endpunkt angehängt ist, gewährt sie allen Prinzipalen auf allen Ressourcen Zugriff auf die aufgelisteten Amazon DocumentDB-API-Aktionen.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Principal": "*",
      "Effect": "Allow",
      "Action": [
        "rds:CreateDBInstance",
        "rds:ModifyDBInstance",
        "rds:CreateDBSnapshot"
      ],
      "Resource": "*"
    }
  ]
}
```

Beispiel: VPC-Endpunktrichtlinie, die jeglichen Zugriff von einem bestimmten Objekt aus verweigert AWS-Konto

Die folgende VPC-Endpunktrichtlinie verweigert AWS-Konto 123456789012 jeglichen Zugriff auf Ressourcen, die den Endpunkt verwenden. Die Richtlinie erlaubt alle Aktionen von anderen. AWS-Konten

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    },
    {
      "Action": "*",
      "Effect": "Deny",
      "Resource": "*",
      "Principal": { "AWS": [ "123456789012" ] }
    }
  ]
}
```

Bewährte Sicherheitsmethoden für Amazon DocumentDB

Amazon DocumentDB bietet Sicherheitsfunktionen wie Amazon VPC-Isolierung, Verschlüsselung im Ruhezustand mithilfe von TLS AWS KMS, Verschlüsselung bei der Übertragung mithilfe von TLS und feingranulare Zugriffskontrolle über IAM. Verwenden Sie die folgenden bewährten Methoden, um Ihre Amazon DocumentDB-Cluster, -Instances und -Daten zu sichern.

Verwenden Sie AWS Identity and Access Management (IAM-) Rollen, um den Zugriff auf Amazon DocumentDB-API-Operationen zu kontrollieren. Dazu gehören Operationen, die Amazon DocumentDB-Ressourcen wie Cluster, Sicherheitsgruppen und Parametergruppen erstellen, ändern oder löschen. Wenn Anwendungen oder andere AWS Dienste auf Amazon DocumentDB-Ressourcen zugreifen müssen, verwenden Sie eine IAM-Rolle, um temporäre Anmeldeinformationen bereitzustellen, anstatt langfristige Anmeldeinformationen in Ihrer Anwendung zu speichern. Wenden Sie beim Erstellen von IAM-Rollen das Prinzip der geringsten Rechte an. Weitere Informationen finden Sie unter IAM-Rollen und Allgemeine Szenarien für Rollen im IAM-Benutzerhandbuch.

- Erzwingen Sie die geringste Berechtigung mit rollenbasierter Zugriffssteuerung.
- Verwenden Sie nicht den AWS-Konto Root-Benutzer, um Amazon DocumentDB-Ressourcen zu verwalten. Verwenden Sie stattdessen AWS Identity and Access Management Identity Center, um menschliche Benutzer mit temporären Anmeldeinformationen zu erstellen. Weitere Informationen finden Sie unter IAM Identity Center-Benutzer im IAM-Benutzerhandbuch.
- Gewähren Sie jeder Identität die Mindestanzahl an Berechtigungen, die zur Erfüllung ihrer Aufgaben erforderlich sind.
- Verwenden Sie IAM-Richtlinien, um Berechtigungen zu verwalten, und folgen Sie dem Prinzip der geringsten Rechte. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.
- Wenn Sie IAM-Benutzer mit langfristigen Anmeldeinformationen verwenden, wechseln Sie diese Anmeldeinformationen regelmäßig.

> **Warning**
>
> Dieses Szenario erfordert IAM-Benutzer mit programmatischem Zugriff und langfristigen Anmeldeinformationen, was ein Sicherheitsrisiko darstellt. Um dieses Risiko zu minimieren, empfehlen wir, diesen Benutzern nur die Berechtigungen zu gewähren, die sie für die Ausführung der Aufgabe benötigen, und diese Benutzer zu entfernen, wenn sie nicht mehr benötigt werden. Die Zugriffsschlüssel können bei Bedarf aktualisiert werden. Weitere Informationen finden Sie unter Zugriffsschlüssel aktualisieren im IAM-Benutzerhandbuch.

- Konfigurieren Sie Secrets Manager so, dass die Secrets für Amazon DocumentDB automatisch rotiert werden. Weitere Informationen finden Sie unter Rotating your Secrets und Rotating Secrets for Amazon DocumentDB im AWS Secrets Manager Benutzerhandbuch.
- Verwenden Sie Transport Layer Security (TLS) und die Verschlüsselung ruhender Daten, um Ihre Daten zu verschlüsseln.

Prüfung von Amazon DocumentDB-Ereignissen

Mit Amazon DocumentDB (mit MongoDB-Kompatibilität) können Sie Ereignisse überprüfen, die in Ihrem Cluster durchgeführt wurden. Beispiele für protokollierte Ereignisse sind erfolgreiche und fehlgeschlagene Authentifizierungsversuche, Drop-Ereignisse für Sammlungen in einer Datenbank oder das Erstellen eines Index. Standardmäßig ist das Auditing in Amazon DocumentDB deaktiviert und erfordert, dass Sie sich für die Nutzung dieser Funktion anmelden.

Wenn die Überwachung aktiviert ist, zeichnet Amazon DocumentDB Ereignisse in den Bereichen Data Definition Language (DDL), Data Manipulation Language (DML), Authentifizierung, Autorisierung und Benutzerverwaltung in Amazon Logs auf. CloudWatch Wenn das Auditing aktiviert ist, exportiert Amazon DocumentDB die Audit-Aufzeichnungen Ihres Clusters (JSON-Dokumente) in Amazon Logs. CloudWatch Sie können Amazon CloudWatch Logs verwenden, um Ihre Amazon DocumentDB-Prüfungsereignisse zu analysieren, zu überwachen und zu archivieren.

Amazon DocumentDB erhebt zwar keine zusätzlichen Kosten für die Aktivierung der Prüfung, aber für die Nutzung von CloudWatch Logs werden Ihnen Standardtarife berechnet. Informationen zu den Preisen für CloudWatch Logs finden Sie unter CloudWatch Amazon-Preise.

Die Amazon DocumentDB-Auditing-Funktion unterscheidet sich deutlich von der Nutzung der Serviceressourcen, mit der überwacht wird. AWS CloudTrail CloudTrail zeichnet Operationen auf, die mit AWS Command Line Interface (AWS CLI) oder AWS-Managementkonsole an Ressourcen wie Clustern, Instances, Parametergruppen und Snapshots ausgeführt werden. Die Überwachung von Ressourcen mit CloudTrail ist standardmäßig aktiviert und kann nicht deaktiviert werden. Die Amazon DocumentDB-Auditing-Funktion ist eine Opt-in-Funktion. Sie zeichnet Operationen auf, die innerhalb Ihres Clusters für Objekte, wie z. B. Datenbanken, Sammlungen, Indizes und Benutzer ausgeführt werden.

Themen

- Unterstützte Ereignisse
- Auditing aktivieren
- Auditing deaktivieren
- Zugriff auf Ihre Prüfereignisse
- Filterung von DML-Audit-Ereignissen

Unterstützte Ereignisse

Amazon DocumentDB Auditing unterstützt die folgenden Ereigniskategorien:

- Data Definition Language (DDL) — umfasst Datenbankverwaltungsvorgänge, Verbindungen, Benutzerverwaltung und Autorisierung.
- Leseereignisse der Data Manipulation Language (DML-Lesevorgänge) — umfasst `find()` und die verschiedenen Aggregationsoperatoren, arithmetischen Operatoren, boolesche Operatoren und andere Operatoren für Leseabfragen.

- Schreibereignisse der Data Manipulation Language (DML-Schreibvorgänge) — schließt ein und Operatoren ein `insert()`, `update()`, `delete()`, `bulkWrite()`

Folgende Ereignistypen werden unterstützt:

Ereignistyp	Kategorie	Description
`authCheck`	Autorisierung	Ergebniscode 0: Erfolg
		Ergebniscode 13: Unautorisierte Versuche, einen Vorgang auszuführen.
`authenticate`	Connection (Verbindung)	Erfolgreiche oder fehlgeschlagene Authentifizierungsversuche bei einer neuen Verbindung.
`auditConfigure`	DDL	Überprüfen Sie die Filterkonfiguration.
`createDatabase`	DDL	Erstellung einer neuen Datenbank.
`createCollection`	DDL	Erstellung einer neuen Sammlung innerhalb einer Datenbank.
`createIndex`	DDL	Erstellung eines neuen Index innerhalb einer Sammlung.

Ereignistyp	Kategorie	Description
`dropCollection`	DDL	Löschen einer Sammlung in einer Datenbank.
`dropDatabase`	DDL	Löschen einer Datenbank.
`dropIndex`	DDL	Löschen eines Index innerhalb einer Sammlung.
`modifyChangeStreams`	DDL	Der Change-Stream wurde erstellt.
`renameCollection`	DDL	Umbenennen einer Sammlung innerhalb einer Datenbank.
`createRole`	Rollenverwaltung	Eine Rolle erstellen.
`dropAllRolesFromDatabase`	Rollenmanagement	Löschen aller Rollen in einer Datenbank.
`dropRole`	Rollenverwaltung	Eine Rolle fallen lassen.
`grantPrivilegesToRole`	Rollenmanagement	Einer Rolle Rechte gewähren.
`grantRolesToRole`	Rollenverwaltung	Einer benutzerd efinierten Rolle Rollen zuweisen.
`revokePrivilegesFromRole`	Verwaltung der Rollen	Entzug von Rechten für eine Rolle.

Ereignistyp	Kategorie	Description
`revokeRolesFromRole`	Rollenverwaltung	Widerruf von Rollen aus einer benutzerd efinierten Rolle.
`updateRole`	Verwaltung von Rollen	Eine Rolle aktualisi eren.
`createUser`	Benutzerverwaltung	Anlegen eines neuen Benutzers.
`dropAllUsersFromDatabase`	Benutzerverwaltung	Löschen aller Benutzer innerhalb einer Datenbank.
`dropUser`	Benutzerverwaltung	Löschen eines bestehenden Benutzers.
`grantRolesToUser`	Benutzerverwaltung	Einem Benutzer Rollen zuweisen.
`revokeRolesFromUser`	Benutzerverwaltung	Einem Benutzer Rollen entziehen.
`updateUser`	UserManagement	Aktualisierung eines bestehenden Benutzers.
`insert`	DML schreiben	Fügt ein Dokument oder mehrere Dokumente in eine Sammlung ein.
`delete`	DML schreiben	Löscht ein Dokument oder mehrere Dokumente aus einer Sammlung.

Ereignistyp	Kategorie	Description
`update`	DML schreiben	Ändert ein vorhanden es Dokument oder Dokumente in einer Sammlung.
`bulkWrite`	DML schreiben	Führt mehrere Schreibvorgänge mit Steuerelementen für die Ausführun gsreihenfolge aus.
`setAuditConfig`	DML schreiben	Richten Sie einen neuen Filter für die DML-Überwachung ein.
`count`	DML gelesen	Gibt die Anzahl der Dokumente zurück, die einer find () -Abfrage für die Sammlung oder Ansicht entsprechen würden.
`countDocuments`	DML gelesen	Gibt die Anzahl der Dokumente zurück, die der Abfrage für eine Sammlung oder Ansicht entsprechen.
`find`	DML gelesen	Wählt Dokumente in einer Sammlung oder Ansicht aus und führt einen Cursor zu den ausgewählten Dokumenten zurück.

Ereignistyp	Kategorie	Description
`getAuditConfig`	DML gelesen	Ruft den aktuellen Filter für die DML-Überwachung ab.
`findAndModify`	DML lesen und DML schreiben	Ändert ein einzelnes Dokument und gibt es zurück.
`findOneAndDelete`	DML lesen und DML schreiben	Löscht ein einzelnes Dokument auf der Grundlage der Filter- und Sortierkr iterien und gibt das gelöschte Dokument zurück.
`findOneAndReplace`	DML lesen und DML schreiben	Ersetzt ein einzelnes Dokument auf der Grundlage des angegebenen Filters.
`findOneAndUpdate`	DML lesen und DML schreiben	Aktualisiert ein einzelnes Dokument auf der Grundlage der Filter- und Sortierkr iterien.
`aggregate`	DML lesen und DML schreiben	Unterstützt APIs in der Aggregationspipeli ne.

Ereignistyp	Kategorie	Description
`distinct`	DML lesen	Findet die unterschiedlichen Werte für ein bestimmtes Feld in einer einzelnen Sammlung oder Ansicht und gibt die Ergebnisse in einem Array zurück.

> **Note**
>
> Für Werte im Parameterfeld des DML-Ereignisdokuments gilt eine Größenbeschränkung von 1 KB. Amazon DocumentDB kürzt den Wert, wenn er 1 KB überschreitet.

> **Note**
>
> TTL-Löschereignisse werden derzeit nicht überwacht.

Auditing aktivieren

Die Aktivierung des Prüfens für einen Cluster ist ein zweistufiger Prozess. Stellen Sie sicher, dass beide Schritte abgeschlossen sind, da sonst keine Audit-Logs an CloudWatch Logs gesendet werden.

Schritt 1. Aktivieren Sie den Clusterparameter audit_logs

Um das Auditing zu aktivieren, müssen Sie den `audit_logs` Parameter in der Parametergruppe ändern. `audit_logs`ist eine kommagetrennte Liste der zu protokollierenden Ereignisse. Ereignisse müssen in Kleinbuchstaben angegeben werden und zwischen den Listenelementen darf kein Leerzeichen stehen.

Sie können die folgenden Werte für die Parametergruppe festlegen:

Wert	Description	
`ddl`	Wenn Sie dies einstellen, wird die Überwachung von DDL-Ereignissen wie CreateDatabase, DropDatabase, CreateCollection, DropCollection, CreateIndex, DropIndex, AuthCheck, authenticate, createUser, dropUser, grant, revoke, updateUser und drop aktiviert RolesToUser RolesFromUser AllUsersFromDatabase	
`dml_read`	Wenn Sie dies einstellen, wird die Prüfung von DML-Leseereignissen wie find, sort count, distinct, group, projecta, unwind, geoNear, GeoIntersects, GeoWithin und anderen MongoDB-Leseabfrageoperatoren ermöglicht.	
`dml_write`	Wenn Sie dies einstellen, wird die	

Wert	Description	
	Überwachung von DML-Schreibereignissen wie insert (), update (), delete () und bulkWrite () aktiviert	
`all`	Wenn Sie dies einstellen, wird die Überwachung Ihrer Datenbankereignisse wie Lese-Abfragen, Schreibabfragen, Datenbankaktionen und Administratoraktionen aktiviert.	
none	Wenn Sie dies einstellen, wird das Auditing deaktiviert	

Wert	Description	
`enabled` (veraltet)	Dies ist eine ältere Parametereinstellung, die 'ddl' entspricht. Wenn Sie dies einstellen, wird die Überwachung von DDL-Ereignissen wie CreateDatabase, DropDatabase, CreateCollection, DropCollection, CreateIndex, DropIndex, AuthCheck, authenticate, createUser, dropUser, grant, revoke, updateUser und drop aktiviert. RolesToUser RolesFromUser AllUsersFromDatabase Wir raten davon ab, diese Einstellung zu verwenden, da es sich um eine ältere Einstellung handelt.	

Wert	Description	
`disabled` (Legacy)	Dies ist eine ältere Parametereinstellung, die „keine“ entsprich t. Wir raten davon ab, diese Einstellung zu verwenden, da es sich um eine ältere Einstellung handelt.	

> **Note**
>
> Der Standardwert für den Clusterparameter audit_logs ist `none` (legacy "`disabled`„).

Sie können die oben genannten Werte auch in Kombinationen verwenden.

Wert	Description	
`ddl, dml_read`	Wenn Sie dies einstellen, wird die Überwachung von DDL-Ereignissen und DML-Leseereignissen aktiviert.	
`ddl, dml_write`	Wenn Sie dies einstellen, wird die Überwachung von DDL-Ereignissen und DML-Schreibvorgäng en aktiviert	
`dml_read, dml_write`	Wenn Sie dies einstellen, wird die Überwachung aller	

Wert	Description	
	DML-Ereignisse aktiviert	

> **Note**
>
> Eine Standardparametergruppe kann nicht abgeändert werden.

Weitere Informationen finden Sie hier:

- Erstellen von Amazon DocumentDB-Cluster-Parametergruppen

  Nach dem Erstellen einer Parametergruppe ändern Sie diese, indem Sie den `audit_logs`-Parameterwert auf `all` ändern.
- Ändern der Amazon DocumentDB-Cluster-Parametergruppen

Schritt 2. Aktivieren Sie den Export von Amazon CloudWatch Logs

Wenn der Wert des `audit_logs` Cluster-Parameters`enabled`,,`ddl`, oder ist`dml_read`, müssen Sie Amazon DocumentDB auch aktivieren`dml_write`, um Protokolle nach Amazon CloudWatch zu exportieren. Wenn Sie einen dieser Schritte auslassen, werden keine Auditprotokolle an gesendet. CloudWatch

Wenn Sie einen Cluster erstellen, eine Point-in-Time-Wiederherstellung durchführen oder einen Snapshot wiederherstellen, können Sie CloudWatch Protokolle aktivieren, indem Sie die folgenden Schritte ausführen.

Using the AWS-Managementkonsole

Informationen zum Aktivieren des Exports von Protokollen durch Amazon DocumentDB zur CloudWatch Verwendung der Konsole finden Sie in den folgenden Themen:

- Beim Erstellen eines Clusters — In Erstellen eines Clusters und einer primären Instance mit dem AWS-Managementkonsole, siehe Cluster erstellen: Zusätzliche Konfigurationen (Schritt 5, Protokollexporte)

- Beim Ändern eines vorhandenen Clusters — Ändern eines Amazon DocumentDB-Clusters
- Bei der Durchführung einer Cluster-Snapshot-Wiederherstellung — Wiederherstellung aus einem Cluster-Snapshot
- Bei der Durchführung einer Point-in-Time-Wiederherstellung — Wiederherstellung auf einen bestimmten Zeitpunkt

Using the AWS CLI

So aktivieren Sie Prüfungsprotokolle beim Erstellen eines neuen Clusters

Mit dem folgenden Code wird der Cluster erstellt `sample-cluster` und CloudWatch Auditprotokolle aktiviert.

Example

Für Linux, macOS oder Unix:

```
aws docdb create-db-cluster \
    --db-cluster-identifier sample-cluster \
    --port 27017 \
    --engine docdb \
    --master-username master-username \
    --master-user-password password \
    --db-subnet-group-name default \
    --enable-cloudwatch-logs-exports audit
```

Für Windows:

```
aws docdb create-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --port 27017 ^
    --engine docdb ^
    --master-username master-username ^
    --master-user-password password ^
    --db-subnet-group-name default ^
    --enable-cloudwatch-logs-exports audit
```

So aktivieren Sie Prüfungsprotokolle beim Ändern eines vorhandenen Clusters

Der folgende Code ändert den Cluster `sample-cluster` und aktiviert CloudWatch Auditprotokolle.

Example

Für Linux, macOS oder Unix:

```
aws docdb modify-db-cluster \
   --db-cluster-identifier sample-cluster \
   --cloudwatch-logs-export-configuration '{"EnableLogTypes":["audit"]}'
```

Für Windows:

```
aws docdb modify-db-cluster ^
   --db-cluster-identifier sample-cluster ^
   --cloudwatch-logs-export-configuration '{"EnableLogTypes":["audit"]}'
```

Die Ausgabe dieser Operationen sieht in etwa wie folgt aus (JSON-Format).

```
{
    "DBCluster": {
        "HostedZoneId": "ZNKXH85TT8WVW",
        "StorageEncrypted": false,
        "DBClusterParameterGroup": "default.docdb4.0",
        "MasterUsername": "<user-name>",
        "BackupRetentionPeriod": 1,
        "Port": 27017,
        "VpcSecurityGroups": [
            {
                "Status": "active",
                "VpcSecurityGroupId": "sg-77186e0d"
            }
        ],
        "DBClusterArn": "arn:aws:rds:us-east-1:900083794985:cluster:sample-cluster",
        "Status": "creating",
        "Engine": "docdb",
        "EngineVersion": "4.0.0",
        "MultiAZ": false,
        "AvailabilityZones": [
            "us-east-1a",
            "us-east-1c",
            "us-east-1f"
        ],
        "DBSubnetGroup": "default",
        "DBClusterMembers": [],
```

```
        "ReaderEndpoint": "sample-cluster.cluster-ro-corcjozrlsfc.us-
east-1.docdb.amazonaws.com",
        "EnabledCloudwatchLogsExports": [
            "audit"
        ],
        "PreferredMaintenanceWindow": "wed:03:08-wed:03:38",
        "AssociatedRoles": [],
        "ClusterCreateTime": "2019-02-13T16:35:04.756Z",
        "DbClusterResourceId": "cluster-YOS52CUXGDTNKDQ7DH72I4LED4",
        "Endpoint": "sample-cluster.cluster-corcjozrlsfc.us-
east-1.docdb.amazonaws.com",
        "PreferredBackupWindow": "07:16-07:46",
        "DBClusterIdentifier": "sample-cluster"
    }
}
```

Auditing deaktivieren

Sie können die Überwachung deaktivieren, indem Sie den CloudWatch Protokollexport und den `audit_logs` Parameter deaktivieren.

Den Export von Protokollen deaktivieren CloudWatch

Sie können den Export von Auditprotokollen deaktivieren, indem Sie entweder die AWS-Managementkonsole oder die AWS CLI verwenden.

Using the AWS-Managementkonsole

Im folgenden Verfahren wird der verwendet AWS-Managementkonsole , um den Export von Protokollen durch Amazon DocumentDB nach zu CloudWatch deaktivieren.

So deaktivieren Sie Prüfungsprotokolle

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Klicken Sie im Navigationsbereich auf Cluster. Wählen Sie anschließend die Schaltfläche links neben dem Namen des Clusters aus, für den Sie das Exportieren von Protokollen deaktivieren möchten.
3. Wählen Sie Actions (Aktionen) und dann Modify (Ändern) aus.

4. Scrollen Sie nach unten zum Abschnitt Log exports (Protokollexporte) und wählen Sie Disabled (Deaktiviert) aus.
5. Klicken Sie auf Weiter.
6. Überprüfen Sie Ihre Änderungen. Wählen Sie anschließend den Zeitpunkt aus, an dem diese Änderung auf Ihren Cluster angewendet werden soll.
   - Apply during the next scheduled maintenance window (Anwendung während des nächsten geplanten Wartungsfensters)
   - Apply immediately (Sofort anwenden)
7. Wählen Sie Cluster bearbeiten aus.

Using the AWS CLI

Der folgende Code ändert den Cluster `sample-cluster` und deaktiviert CloudWatch Audit-Logs.

Example

Für Linux, macOS oder Unix:

```
aws docdb modify-db-cluster \
   --db-cluster-identifier sample-cluster \
   --cloudwatch-logs-export-configuration '{"DisableLogTypes":["audit"]}'
```

Für Windows:

```
aws docdb modify-db-cluster ^
   --db-cluster-identifier sample-cluster ^
   --cloudwatch-logs-export-configuration '{"DisableLogTypes":["audit"]}'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    "DBCluster": {
        "DBClusterParameterGroup": "default.docdb4.0",
        "HostedZoneId": "ZNKXH85TT8WVW",
        "MasterUsername": "<user-name>",
        "Status": "available",
        "Engine": "docdb",
        "Port": 27017,
```

```
        "AvailabilityZones": [
            "us-east-1a",
            "us-east-1c",
            "us-east-1f"
        ],
        "EarliestRestorableTime": "2019-02-13T16:35:50.387Z",
        "DBSubnetGroup": "default",
        "LatestRestorableTime": "2019-02-13T16:35:50.387Z",
        "DBClusterArn": "arn:aws:rds:us-east-1:900083794985:cluster:sample-
cluster2",
        "Endpoint": "sample-cluster2.cluster-corcjozrlsfc.us-
east-1.docdb.amazonaws.com",
        "ReaderEndpoint": "sample-cluster2.cluster-ro-corcjozrlsfc.us-
east-1.docdb.amazonaws.com",
        "BackupRetentionPeriod": 1,
        "EngineVersion": "4.0.0",
        "MultiAZ": false,
        "ClusterCreateTime": "2019-02-13T16:35:04.756Z",
        "DBClusterIdentifier": "sample-cluster2",
        "AssociatedRoles": [],
        "PreferredBackupWindow": "07:16-07:46",
        "DbClusterResourceId": "cluster-YOS52CUXGDTNKDQ7DH72I4LED4",
        "StorageEncrypted": false,
        "PreferredMaintenanceWindow": "wed:03:08-wed:03:38",
        "DBClusterMembers": [],
        "VpcSecurityGroups": [
            {
                "Status": "active",
                "VpcSecurityGroupId": "sg-77186e0d"
            }
        ]
    }
}
```

Deaktivierung des Parameters audit_logs

Um den Parameter `audit_logs` für Ihren Cluster zu deaktivieren, können Sie den Cluster so ändern, dass er eine Parametergruppe mit dem Wert `disabled` für den Parameter `audit_logs` verwendet. Sie können auch den Wert des Parameters `audit_logs` in der Parametergruppe des Clusters in `disabled` ändern.

Weitere Informationen finden Sie unter den folgenden Themen:

- Ändern eines Amazon DocumentDB-Clusters
- Ändern der Amazon DocumentDB-Cluster-Parametergruppen

Zugriff auf Ihre Prüfereignisse

Gehen Sie wie folgt vor, um auf Ihre Audit-Ereignisse bei Amazon zuzugreifen CloudWatch.

1. Öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Stellen Sie sicher, dass Sie sich in derselben Region wie Ihr Amazon DocumentDB-Cluster befinden.
3. Wählen Sie im Navigationsbereich Protokolle aus.
4. Um die Prüfprotokolle für Ihren Cluster zu finden, suchen Sie in der Liste und wählen Sie **`/aws/docdb/`*`yourClusterName`*`/audit`** aus.

   Die Prüfereignisse für Ihre Instances sind unter dem jeweiligen Instance-Namen verfügbar.

Filterung von DML-Audit-Ereignissen

Erste Schritte mit der DML-Auditfilterung

DML-Auditereignisse können gefiltert werden, bevor sie an Amazon geschrieben werden. CloudWatch Um diese Funktion nutzen zu können, müssen Audit-Log und DML-Logging aktiviert sein. Amazon DocumentDB unterstützt das Filtern nach `atypecommand`, `usernamespace`, und. `auditAuthorizationSuccess`

> **Note**
>
> DDL-Ereignisse werden nicht gefiltert.

Sie können die Überwachungsfilterung jederzeit aktivieren, indem Sie den Überwachungsfilter mithilfe der `auditAuthorizationSuccess` Parameter `setAuditConfigfilter`, und in der `db.adminCommand( { command } )` Operation angeben:

```
db.admin.runCommand(
   {
```

```
      setAuditConfig: 1,
      filter:
         {
            //filter conditions
         },
      auditAuthorizationSuccess: true | false
   }
)
```

Sie können die Überwachungsfiltereinstellungen auch abrufen, indem Sie den folgenden Befehl ausführen:

```
db.admin.runCommand( { getAuditConfig: 1})
```

Sicherheitsanforderungen

Nur Datenbanken users/roles mit berechtigter Aktion `auditConfigure` können die oben genannten Befehle ausführen, `admindb` wenn DML-Auditfilter gesetzt oder aufgelistet werden. Sie können entweder eine der integrierten Rollen von [`clusterAdmin`,`hostManager`,`root`] verwenden oder benutzerdefinierte Rollen mit `auditConfigure` Rechten erstellen. Im Folgenden finden Sie ein Beispiel für die Verwendung vorhandener Rollen mit dieser `auditConfigure` Berechtigung und ein Beispiel für benutzerdefinierte Rollen.

Benutzer mit integrierter Rolle:

```
use admin
db.createUser(
  {
    user: "myClusterAdmin",
    pwd: "password123",
    roles: [ { role: "clusterAdmin", db: "admin" } ]
  }
)
```

Benutzer mit benutzerdefinierten Rollen:

```
use admin
db.createRole(
   {
     role: "myRole",
```

```
     privileges: [
       { resource: { cluster: true }, actions: [ "auditConfigure" ] }
     ],
     roles: []
   }
)
db.createUser(
  {
    user: "myUser",
    pwd: "myPassword",
    roles: [ { role: "myRole", db: "admin" } ]
  }
)
```

Anwendungsfälle filtern

Beispiel: Ereignisse anhand von Befehlen filtern

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {
        "$and": [
         {
            "param.command":
               {
                  $in: [ "find","count", "insert", "delete", "update",
 "findandmodify" ]
               }
         }
         ]
      },
      auditAuthorizationSuccess: true
   }
)
```

Beispiel: Ereignisse nach Benutzernamen filtern

In diesem Beispiel wird nur der Benutzer „myUser“ protokolliert:

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
```

```
      filter: {
      "$and": [
         {
            "param.user":
               {
                  $in: [ "myUser" ]
               }
         }
         ]},
      auditAuthorizationSuccess: true})
```

Beispiel: Filtern nach **`atype`**

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {atype: "authCheck"},
      auditAuthorizationSuccess: true
   })
```

> **Note**
>
> Alle DML-Protokolle haben `authCheck` als`atype`. Nur DDL hat eine andere. `atype` Wenn Sie einen anderen Wert als `authCheck` in eingeben`filter`, wird keine DML-Anmeldung generiert. CloudWatch

Beispiel: Filtern mithilfe mehrerer Filter, die durch Operatoren verbunden sind

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {
      "$and": [
         {
            "param.command":
               {
                  $in: [ "find","count", "insert", "delete", "update",
 "findandmodify" ]
               }
         }
```

```
        ],
       "$nor": [
        {
           "param.command":
              {
                 $in: ["count", "insert", "delete", "update", "findandmodify" ]
              }
        }]
       },
      auditAuthorizationSuccess: true})
```

> **Note**
>
> Auf der obersten Ebene `$nor` werden nur `$and``$or`, und unterstützt. Alle anderen Operatoren werden nicht unterstützt und führen zu einem Fehler.

Beispiel: Filtern nach Ereignissen nach **`auditAuthorizationSuccess`**

In diesem Filter werden alle Befehle, die die Autorisierung erfolgreich bestanden haben, nicht protokolliert:

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {},
      auditAuthorizationSuccess: false
   }
)
```

Beispiel: Filtern mit **`$in`** und **`$nin`** Bedingungen

Wenn Sie `$in` sowohl in als auch verwenden`$nin`, wird der Befehl nicht protokolliert, da zwischen den Bedingungen ein implizites „und“ steht. In diesem Beispiel blockiert Regex den `find` Befehl, sodass nichts protokolliert wird:

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {
      "$and": [
```

```
         {
            atype: "authCheck",
            "param.command":
               {
                  $in: [ "find", "insert", "delete", "update", "findandmodify" ],
                  $nin: ["count", "insert", "delete", "update", "findandmodify" ],
                  $not: /^^find.*/
               }
         },
         ],
       "$or": [
         {
            "param.command":
               {
                  $nin: ["count", "insert", "delete", "update", "findandmodify" ]
               }
         }]
       },
      auditAuthorizationSuccess: true})
```

Beispiel: Filtern nach **`namespace`**

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {
      "$and": [
         {
            "param.ns":
               {
                  $in: [ "test.foo" ]
               }
         }
         ]},
      auditAuthorizationSuccess: true})
```

Beispiel: Auf Standardfilter zurücksetzen

Das Zurücksetzen auf den Standardwert bedeutet, dass jedes DML-Audit-Ereignis protokolliert wird. Führen Sie den folgenden Befehl aus, um die Filterung auf den Standardwert zurückzusetzen:

```
db.admin.runCommand(
   {
```

```
      setAuditConfig: 1,
      filter: {},
      auditAuthorizationSuccess: true
   }
)
```

Amazon VPC und Amazon DocumentDB

Amazon Virtual Private Cloud (Amazon VPC) ermöglicht es Ihnen, AWS Ressourcen wie Amazon DocumentDB-Instances in einer Virtual Private Cloud (VPC) zu starten.

Wenn Sie eine VPC verwenden, haben Sie die Kontrolle über Ihre virtuelle Netzwerkumgebung. Sie können Ihren eigenen IP-Adressbereich auswählen, Subnetze erstellen sowie Routing-Tabellen und Zugriffskontrolllisten konfigurieren. Für den Betrieb Ihres Clusters in einer VPC fallen keine zusätzlichen Kosten an.

Konten haben eine Standard-VPC. Alle neuen Cluster werden in der Standard-VPC erstellt, sofern Sie nichts anderes angeben.

Topics

- DocumentDB-Cluster in einer VPC
- Zugreifen auf einen Amazon DocumentDB-Cluster in einer VPC
- Erstellen Sie eine IPv4-only VPC zur Verwendung mit einem DocumentDB-Cluster
- Erstellen Sie eine Dual-Stack-VPC zur Verwendung mit einem DocumentDB-Cluster

Im Folgenden finden Sie eine Diskussion über VPC-Funktionen, die für Amazon DocumentDB-Cluster relevant sind. Weitere Informationen zur Amazon VPC-Sicherheit finden Sie unter Sicherheit im Amazon VPC-Benutzerhandbuch.

DocumentDB-Cluster in einer VPC

Ihr Amazon DocumentDB-Cluster befindet sich in einer Virtual Private Cloud (VPC). Eine VPC ist ein virtuelles Netzwerk, das logisch von anderen virtuellen Netzwerken in der isoliert ist. AWS Cloud Amazon VPC ermöglicht es Ihnen, AWS Ressourcen wie einen Amazon DocumentDB-Cluster oder eine Amazon EC2-Instance in einer VPC zu starten. Bei der VPC kann es sich um die mit Ihrem Konto verknüpfte Standard-VPC oder eine von Ihnen erstellte VPC handeln. Alle VPCs sind mit Ihrem Konto verknüpft. AWS

Die Standard-VPC besitzt drei Subnetze, mit denen Sie Ressourcen innerhalb der VPC separieren können. Zudem verfügt die Standard-VPC über ein Internet-Gateway, mit dem Sie den externen Zugriff auf Ressourcen in der VPC gewähren können.

Szenarien mit Amazon DocumentDB-Clustern in einer VPC und außerhalb einer VPC finden Sie unter. Zugreifen auf einen Amazon DocumentDB-Cluster in einer VPC

Topics

- Arbeiten mit Clustern in einer VPC
- Arbeiten mit Subnetzgruppen
- Amazon DocumentDB-IP-Adressierung
- Erstellen eines Clusters in einer VPC

Arbeiten mit Clustern in einer VPC

Hier sind einige Tipps zum Arbeiten mit einem Cluster in einer VPC:

- Ihre VPC muss mindestens zwei Subnetze besitzen. Diese Subnetze müssen sich in zwei verschiedenen Availability Zones befinden, in AWS-Region denen Sie Ihren Cluster bereitstellen möchten. Ein Subnetz ist ein Segment des IP-Adressbereichs einer VPC, das Sie angeben können und das Sie verwenden können, um Cluster basierend auf Ihren Sicherheits- und Betriebsanforderungen zu gruppieren.
- Wenn Sie möchten, dass Ihr Cluster in der VPC öffentlich zugänglich ist, stellen Sie sicher, dass Sie die VPC-Attribute und aktivieren. `DNS hostnames` `DNS resolution`
- Ihre VPC muss über eine Subnetzgruppe verfügen, die Sie erstellen. Sie erstellen eine Subnetzgruppe, indem Sie die Subnetze angeben, die Sie erstellt haben. Amazon DocumentDB wählt ein Subnetz und eine IP-Adresse innerhalb dieser Subnetzgruppe aus, um sie der primären Instance in Ihrem Cluster zuzuordnen. Die primäre Instance verwendet die Availability Zone, die das Subnetz enthält.
- Ihre VPC muss über eine VPC-Sicherheitsgruppe verfügen, die den Zugriff auf den Cluster ermöglicht.

  Weitere Informationen finden Sie unter Zugreifen auf einen Amazon DocumentDB-Cluster in einer VPC.
- Die CIDR-Blöcke in jedem Ihrer Subnetze müssen groß genug sein, um freie IP-Adressen aufzunehmen, die Amazon DocumentDB bei Wartungsarbeiten, einschließlich Failover und

Rechenskalierung, verwenden kann. Zum Beispiel ein Bereich wie 10.0.0. 0/24 und 10.0.1. 0/24 ist normalerweise groß genug.

- Eine VPC kann das `instance tenancy` Attribut entweder `default` oder `dedicated` haben. Bei allen Standard-VPCs ist das `instance tenancy` Attribut auf Standard festgelegt, und eine Standard-VPC kann jede Instanzklasse unterstützen.

  Wenn Sie sich dafür entscheiden, Ihren Cluster in einer dedizierten VPC zu haben, auf der das `instance tenancy` Attribut auf festgelegt ist`dedicated`, muss es sich bei der Instance-Klasse Ihres Clusters um einen der zugelassenen Amazon EC2-Instance-Typen handeln. Die dedizierte `r5.large` EC2-Instance entspricht beispielsweise der `db.r5.large` Instance-Klasse. Informationen zur Instance-Tenancy in einer VPC finden Sie unter Amazon EC2-Instances im Amazon Elastic Compute Cloud-Benutzerhandbuch.

  Weitere Informationen zu Instance-Typen, die in einer Dedicated Instance enthalten sein dürfen, finden Sie unter Amazon EC2 Dedicated Instances auf der Preisseite für Amazon EC2.

  > **Note**
  >
  > Wenn Sie das `instance tenancy` Attribut `dedicated` für einen Cluster auf setzen, garantiert dies nicht, dass der Cluster auf einem dedizierten Host ausgeführt wird.

Arbeiten mit Subnetzgruppen

Subnetze sind Segmente eines IP-Adressbereichs der VPC, die Sie festlegen und mit denen Sie basierend auf Ihren Sicherheits- und Betriebsanforderungen Ressourcen gruppieren können. Eine Subnetzgruppe ist eine Sammlung von Subnetzen (in der Regel privat), die Sie in einer VPC erstellen und die Sie dann für Ihre Cluster festlegen. Mithilfe einer Subnetzgruppe können Sie eine bestimmte VPC angeben, wenn Sie Cluster mithilfe der oder der Amazon DocumentDB-API erstellen. AWS CLI Wenn Sie die Konsole verwenden, können Sie die VPC und Subnetze auswählen, die Sie verwenden möchten.

Jede Subnetzgruppe sollte Subnetze in mindestens zwei Availability Zones in einer bestimmten Umgebung haben. AWS-Region Wenn Sie einen Cluster in einer VPC erstellen, wählen Sie eine Subnetzgruppe dafür aus. Aus der Subnetzgruppe wählt Amazon DocumentDB ein Subnetz und eine IP-Adresse innerhalb dieses Subnetzes aus, um sie der primären Instance in Ihrem Cluster zuzuordnen. Die Datenbank verwendet die Availability Zone, die das Subnetz enthält. DocumentDB weist immer eine IP-Adresse aus einem Subnetz zu, das freien IP-Adressraum hat.

Die Subnetze in einer Subnetzgruppe sind entweder öffentlich oder privat, je nachdem, welche Konfiguration Sie für ihre Netzwerkzugriffskontrolllisten (Netzwerk-ACLs) und Routingtabellen festgelegt haben. Damit ein Cluster öffentlich zugänglich ist, müssen alle Subnetze in seiner Subnetzgruppe öffentlich sein. Wenn ein Subnetz, das einem öffentlich zugänglichen Cluster zugeordnet ist, von öffentlich zu privat wechselt, kann dies die Clusterverfügbarkeit beeinträchtigen.

Um eine Subnetzgruppe zu erstellen, die den Dual-Stack-Modus unterstützt, stellen Sie sicher, dass jedem Subnetz, das Sie der Subnetzgruppe hinzufügen, ein IPv6 (Internet Protocol Version 6) CIDR-Block zugeordnet ist. Weitere Informationen finden Sie unter Amazon DocumentDB-IP-Adressierung IPv6-Unterstützung für Ihre VPC im Amazon Virtual Private Cloud-Benutzerhandbuch.

Wenn Amazon DocumentDB einen Cluster in einer VPC erstellt, weist es dem Cluster mithilfe einer IP-Adresse aus Ihrer Subnetzgruppe eine Netzwerkschnittstelle zu. Verwenden Sie immer den DNS-Namen (Domain Name System), um eine Verbindung zu Ihrem Cluster herzustellen, da sich die zugrunde liegende IP-Adresse während des Failovers ändert.

> **Note**
>
> Stellen Sie sicher, dass Sie für jeden Cluster, den Sie in einer VPC ausführen, mindestens eine Adresse in jedem Subnetz der Subnetzgruppe reservieren, die Amazon DocumentDB für Wiederherstellungsaktionen verwenden kann.

Gemeinsam genutzte Subnetze

Sie können einen Cluster in einer gemeinsam genutzten VPC erstellen.

Einige Aspekte, die Sie bei der Verwendung gemeinsam genutzter VPCs beachten sollten:

- Sie können einen Cluster von einem gemeinsam genutzten VPC-Subnetz in ein nicht gemeinsam genutztes VPC-Subnetz verschieben und umgekehrt.
- Teilnehmer einer gemeinsam genutzten VPC müssen in der VPC eine Sicherheitsgruppe erstellen, damit sie einen Cluster erstellen können.
- Besitzer und Teilnehmer einer gemeinsam genutzten VPC können mithilfe von DocumentDB-Abfragen auf die Datenbank zugreifen. Allerdings kann nur der Ersteller einer Ressource beliebige API-Aufrufe für die Ressource tätigen.

Amazon DocumentDB-IP-Adressierung

IP-Adressen ermöglichen es Ressourcen in Ihrer VPC untereinander und mit Ressourcen im Internet zu kommunizieren. Amazon DocumentDB unterstützt sowohl IPv4- als auch IPv6-Adressierungsprotokolle. Standardmäßig verwenden Amazon DocumentDB und Amazon VPC das IPv4-Adressierungsprotokoll. Sie können dieses Standardverhalten nicht deaktivieren. Wenn Sie eine VPC erstellen, müssen Sie einen IPv4 CIDR-Block (einen privaten IPv4-Adressenbereich) angeben. Sie können Ihrer VPC und Ihren Subnetzen optional einen IPv6-CIDR-Block zuweisen und IPv6-Adressen aus diesem Block Clustern in Ihrem Subnetz zuweisen.

> **Note**
>
> Dual-stack Der Modus (IPv6-Adressierung) wird nur in den Amazon DocumentDB-Versionen 4.0 und 5.0 unterstützt.

Durch die Unterstützung des IPv6-Protokolls wird die Anzahl der unterstützten IP-Adressen erweitert. Durch die Verwendung des IPv6-Protokolls stellen Sie sicher, dass Sie über ausreichende verfügbare Adressen für das künftige Wachstum des Internets verfügen. Neue und vorhandene DocumentDB-Ressourcen können IPv4- und IPv6-Adressen in Ihrer VPC verwenden. Das Konfigurieren, Sichern und Übersetzen des Netzwerkverkehrs zwischen den beiden Protokollen, die in verschiedenen Teilen einer Anwendung verwendet werden, können den Betriebsaufwand erhöhen. Sie können das IPv6-Protokoll für Amazon DocumentDB-Ressourcen standardisieren, um Ihre Netzwerkkonfiguration zu vereinfachen.

Topics

- IPv4-Adressen
- IPv6-Adressen
- Dual-stack Modus

IPv4-Adressen

Beim Erstellen einer VPC müssen Sie für diese einen IPv4-Adressbereich in Form eines CIDR-Blocks wie `10.0.0.0/16` festlegen. Eine Subnetzgruppe definiert den Bereich der IP-Adressen in diesem CIDR-Block, den ein Cluster verwenden kann. Diese IP-Adressen können privat oder öffentlich sein.

Eine private IPv4-Adresse ist eine IP-Adresse, die nicht über das Internet erreichbar ist. Sie können private IPv4-Adressen für die Kommunikation zwischen Ihrem Cluster und anderen Ressourcen, wie

Amazon EC2-Instances, in derselben VPC verwenden. Jeder Cluster hat eine private IP-Adresse für die Kommunikation in der VPC.

Eine öffentliche IP-Adresse ist eine IPv4-Adresse, die über das Internet erreichbar ist. Öffentliche IP-Adressen sind für Ihren DocumentDB-Cluster nicht zulässig. Jede öffentliche IP-Adresse sollte vom Internet-Gateway und von EC2 im öffentlichen Subnetz aufgelöst werden.

Informationen zum Erstellen einer VPC mit ausschließlich privaten IPv4-Adressen, die Sie für ein gängiges Amazon DocumentDB-Szenario verwenden können, finden Sie unter. Erstellen Sie eine IPv4-only VPC zur Verwendung mit einem DocumentDB-Cluster

IPv6-Adressen

Optional können Sie Ihrer VPC und den Subnetzen einen IPv6 CIDR-Block zuordnen und den Ressourcen in Ihrer VPC IPv6-Adressen von diesem Block zuweisen. Jede IPv6-Adresse ist global eindeutig.

Der IPv6 CIDR-Block für Ihre VPC wird automatisch aus dem Amazon-Pool mit IPv6-Adressen zugewiesen. Sie können den Bereich nicht selbst auswählen.

Stellen Sie beim Herstellen einer Verbindung mit einer IPv6-Adresse sicher, dass die folgenden Bedingungen erfüllt sind:

- Der Client ist so konfiguriert, dass der Datenverkehr zwischen Client und Datenbank über IPv6 erlaubt ist.
- Die vom Cluster verwendeten DocumentDB-Sicherheitsgruppen sind korrekt konfiguriert, sodass der Datenverkehr von Client zu Datenbank über IPv6 zulässig ist.
- Der Client-Betriebssystemstapel lässt den Datenverkehr auf der IPv6-Adresse zu, und die Betriebssystemtreiber und -Bibliotheken sind so konfiguriert, dass sie den richtigen Standard-Cluster-Endpunkt (entweder IPv4 oder IPv6) auswählen.

Weitere Informationen zu IPv6 finden Sie unter IP-Adressierung im Amazon Virtual Private Cloud-Benutzerhandbuch.

Dual-stack Modus

Wenn ein Cluster sowohl über das IPv4- als auch über das IPv6-Adressierungsprotokoll kommunizieren kann, wird er im Dual-Stack-Modus ausgeführt. Ressourcen können also über IPv4, IPv6 oder beides mit dem Cluster kommunizieren. DocumentDB deaktiviert den Internet-Gateway-Zugriff für IPv6-Endpunkte von privaten Clustern im Dual-Stack-Modus. DocumentDB tut dies, um

sicherzustellen, dass Ihre IPv6-Endpunkte privat sind und nur von Ihrer VPC aus zugegriffen werden kann.

Topics

- Dual-stack Modus- und Subnetzgruppen
- Arbeiten mit Clustern im Dual-Stack-Modus
- IPv4-only Cluster so ändern, dass sie den Dual-Stack-Modus verwenden
- Dual-stack Modus Regions- und Versionsverfügbarkeit
- Einschränkungen für Dual-Stack-Netzwerkcluster

Dual-stack Modus- und Subnetzgruppen

Um den Dual-Stack-Modus zu verwenden, stellen Sie sicher, dass jedem Subnetz in der Subnetzgruppe, die Sie dem Cluster zuordnen, ein IPv6-CIDR-Block zugeordnet ist. Sie können eine neue Subnetzgruppe erstellen oder eine vorhandene Subnetzgruppe ändern, um diese Anforderung zu erfüllen. Nachdem sich ein Cluster im Dual-Stack-Modus befindet, können Clients normal eine Verbindung zu ihm herstellen. Stellen Sie sicher, dass die Client-Sicherheitsfirewalls und die DocumentDB-Cluster-Sicherheitsgruppen korrekt konfiguriert sind, um Datenverkehr über IPv6 zuzulassen. Um eine Verbindung herzustellen, verwenden die Clients den Endpunkt des Clusters. Clientanwendungen können angeben, welches Protokoll bevorzugt wird, wenn eine Verbindung mit einer Datenbank hergestellt wird. Im Dual-Stack-Modus erkennt der Cluster das bevorzugte Netzwerkprotokoll des Clients, entweder IPv4 oder IPv6, und verwendet dieses Protokoll für die Verbindung.

Wenn eine Subnetzgruppe den Dual-Stack-Modus aufgrund einer Subnetzlöschung oder einer CIDR-Trennung nicht mehr unterstützt, besteht das Risiko eines inkompatiblen Netzwerkstatus für Cluster, die der Subnetzgruppe zugeordnet sind. Außerdem können Sie die Subnetzgruppe nicht verwenden, wenn Sie einen neuen Cluster im Dual-Stack-Modus erstellen.

Um mithilfe von zu ermitteln, ob eine Subnetzgruppe den Dual-Stack-Modus unterstützt AWS-Managementkonsole, sehen Sie sich den Netzwerktyp auf der Detailseite der Subnetzgruppe an. Um mithilfe von zu ermitteln, ob eine Subnetzgruppe den Dual-Stack-Modus unterstützt AWS CLI, führen Sie den `describe-db-subnet-groups` Befehl aus und zeigen Sie ihn in der Ausgabe an. `SupportedNetworkTypes`

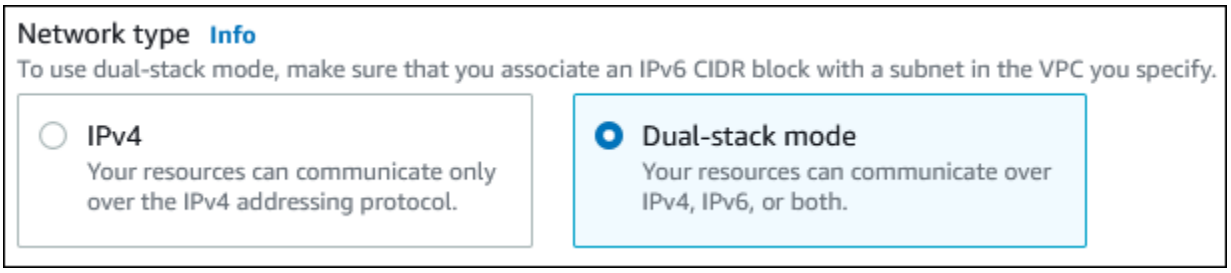
Read Replicas werden als unabhängige Cluster behandelt und können einen anderen Netzwerktyp als der primäre Cluster haben. Wenn Sie den Netzwerktyp des primären Clusters eines Read

Replicas ändern, ist das Read Replica nicht betroffen. Wenn Sie einen Cluster wiederherstellen, können Sie ihn auf jedem unterstützten Netzwerktyp wiederherstellen.

Arbeiten mit Clustern im Dual-Stack-Modus

Wenn Sie einen Cluster erstellen oder ändern, können Sie den Dual-Stack-Modus angeben, damit Ihre Ressourcen über IPv4, IPv6 oder beides mit Ihrem Cluster kommunizieren können.

Wenn Sie den verwenden AWS-Managementkonsole , um einen Cluster zu erstellen oder zu ändern, können Sie den Dual-Stack-Modus im Abschnitt Netzwerktyp angeben. Die folgende Abbildung zeigt den Abschnitt Netzwerktyp in der Konsole:



Wenn Sie den verwenden AWS CLI , um einen Cluster zu erstellen oder zu ändern, stellen Sie die `--network-type` Option `DUAL` auf den Dual-Stack-Modus ein. Wenn Sie die DocumentDB-API verwenden, um einen Cluster zu erstellen oder zu ändern, legen Sie den `NetworkType` Parameter so fest, dass der `DUAL` Dual-Stack-Modus verwendet werden soll. Wenn der Dual-Stack-Modus von der angegebenen DocumentDB-Engine-Version oder Subnetzgruppe nicht unterstützt wird, wird der Fehler zurückgegeben. `NetworkTypeNotSupported`

Weitere Hinweise zum Erstellen eines Clusters finden Sie unter. Erstellen eines Amazon DocumentDB-Clusters Weitere Hinweise zum Ändern eines Clusters finden Sie unterÄndern eines Amazon DocumentDB-Clusters.

Um mithilfe der Konsole festzustellen, ob sich ein Cluster im Dual-Stack-Modus befindet, sehen Sie sich den Netzwerktyp auf der Registerkarte Konnektivität und Sicherheit für den Cluster an.

IPv4-only Cluster so ändern, dass sie den Dual-Stack-Modus verwenden

Sie können einen IPv4-only Cluster so ändern, dass er den Dual-Stack-Modus verwendet. Ändern Sie dazu den Netzwerktyp des Clusters.

Es wird empfohlen, den Netzwerktyp Ihres Amazon DocumentDB-Clusters während eines Wartungsfensters zu ändern. Sie können den Netzwerktyp manuell festlegen, indem Sie den `modify-db-cluster` Befehl verwenden.

Bevor Sie einen Cluster so ändern, dass er den Dual-Stack-Modus verwendet, stellen Sie sicher, dass seine Subnetzgruppe den Dual-Stack-Modus unterstützt. Wenn die dem Cluster zugeordnete Subnetzgruppe den Dual-Stack-Modus nicht unterstützt, geben Sie bei der Änderung des Clusters eine andere Subnetzgruppe an, die ihn unterstützt. Das Ändern der Subnetzgruppe eines Clusters kann zu Ausfallzeiten führen.

Wenn Sie die Subnetzgruppe eines Clusters ändern, bevor Sie den Cluster auf den Dual-Stack-Modus umstellen, stellen Sie sicher, dass die Subnetzgruppe vor und nach der Änderung für den Cluster gültig ist.

Führen Sie den `ModifyDBCluster` Aufruf nur mit dem `NetworkType` Parameter auf aus, um das Netzwerk in den `DUAL` Dual-Stack-Modus zu versetzen. Das Hinzufügen anderer Parameter zusammen mit `NetworkType` demselben API-Aufruf kann zu Ausfallzeiten führen. Wenn Sie mehrere Parameter ändern möchten, stellen Sie sicher, dass die Änderung des Netzwerktyps erfolgreich abgeschlossen wurde, bevor Sie eine weitere `ModifyDBCluster`-Anforderung mit anderen Parametern senden.

Wenn Sie nach der Änderung keine Verbindung zum Cluster herstellen können, stellen Sie sicher, dass die Client- und Datenbank-Sicherheitsfirewalls und Routing-Tabellen korrekt konfiguriert sind, um den Datenverkehr zur Datenbank im ausgewählten Netzwerk (entweder IPv4 oder IPv6) zuzulassen. Möglicherweise müssen Sie auch Betriebssystemparameter, Bibliotheken oder Treiber ändern, um eine Verbindung mithilfe einer IPv6-Adresse herzustellen.

Um einen IPv4-only Cluster so zu ändern, dass er den Dual-Stack-Modus verwendet

1. Ändern Sie eine Subnetzgruppe so, dass sie den Dual-Stack-Modus unterstützt, oder erstellen Sie eine Subnetzgruppe, die den Dual-Stack-Modus unterstützt:

   a. Ordnen Sie Ihrer VPC einen IPv6-CIDR-Block zu.

      Anweisungen finden Sie unter Hinzufügen oder Entfernen eines CIDR-Blocks zu Ihrer VPC im Amazon VPC-Benutzerhandbuch.

   b. Hängen Sie den IPv6-CIDR-Block an alle Subnetze in Ihrer Subnetzgruppe an.

      Anweisungen finden Sie unter Hinzufügen oder Entfernen eines IPv6-CIDR-Blocks zu Ihrem Subnetz im Amazon Virtual Private Cloud-Benutzerhandbuch.

   c. Vergewissern Sie sich, dass die Subnetzgruppe den Dual-Stack-Modus unterstützt.

      Wenn Sie die verwenden AWS-Managementkonsole, wählen Sie die Subnetzgruppe aus und stellen Sie sicher, dass der Wert Unterstützte Netzwerktypen auf Dual gesetzt ist.

Wenn Sie den verwenden AWS CLI, führen Sie den `describe-db-subnet-groups` Befehl aus und stellen Sie sicher, dass der `SupportedNetworkType` Wert für den Cluster lautet`Dual`.

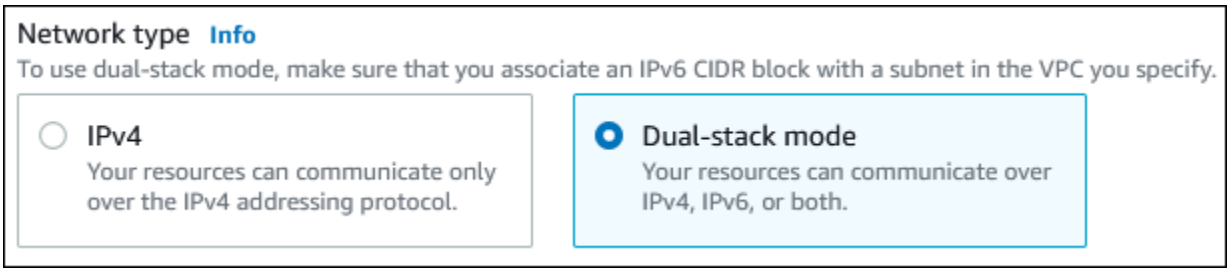
2. Ändern Sie die dem Cluster zugeordnete Sicherheitsgruppe, um IPv6-Verbindungen zur Datenbank zuzulassen, oder erstellen Sie eine neue Sicherheitsgruppe, die IPv6-Verbindungen zulässt.

   Anweisungen finden Sie unter Sicherheitsgruppenregeln im Amazon Virtual Private Cloud-Benutzerhandbuch.

3. Ändern Sie den Cluster so, dass er den Dual-Stack-Modus unterstützt. Stellen Sie dazu den Netzwerktyp auf Dual-stack Modus ein.

   Wenn Sie die Konsole verwenden, stellen Sie sicher, dass die folgenden Einstellungen korrekt sind:

   - Netzwerktyp — Dual-stack Modus

   - Subnetzgruppe — Die Subnetzgruppe, die Sie in einem vorherigen Schritt konfiguriert haben
   - Sicherheitsgruppe — Die Sicherheit, die Sie in einem vorherigen Schritt konfiguriert haben

   Wenn Sie die verwenden AWS CLI, stellen Sie sicher, dass die folgenden Einstellungen korrekt sind:

   - `--network-type` — `dual`
   - `--db-subnet-group-name`— Die Subnetzgruppe, die Sie in einem vorherigen Schritt konfiguriert haben
   - `--vpc-security-group-ids`— Die VPC-Sicherheitsgruppe, die Sie in einem vorherigen Schritt konfiguriert haben

   Beispiel:

```
aws docdb modify-db-cluster --db-cluster-identifier <cluster-name> --network-type
 "DUAL"
```

4. Vergewissern Sie sich, dass der Cluster den Dual-Stack-Modus unterstützt.

   Wenn Sie die Konsole verwenden, wählen Sie die Registerkarte Konnektivität und Sicherheit für den Cluster und stellen Sie sicher, dass der Wert Netzwerktyp auf Dual-stack Modus gesetzt ist.

   Wenn Sie den verwenden AWS CLI, führen Sie den `describe-db-cluster` Befehl aus und stellen Sie sicher, dass der `NetworkType` Wert für den Cluster lautet`dual`.

   Führen Sie den `dig` Befehl auf dem Cluster-Endpunkt aus, um die ihm zugeordnete IPv6-Adresse zu identifizieren:

   ```
   dig <db-cluster-endpoint> AAAA
   ```

   Verwenden Sie den Cluster-Endpunkt, nicht die IPv6-Adresse, um eine Verbindung zum Cluster herzustellen.

Dual-stack Modus Regions- und Versionsverfügbarkeit

Die Verfügbarkeit der Funktionen und der Support variieren je nach Gerät AWS-Regionen.

Regionsunterstützung

Der Dual-Stack-Modus für Amazon DocumentDB ist derzeit in allen kommerziellen AWS-Regionen Versionen verfügbar.

Versionsunterstützung

Dual-stack Der Modus wird in den Amazon DocumentDB-Versionen 4.0, 5.0 und 8.0 unterstützt. Wenn Sie in keiner dieser Versionen auf den Dual-Stack-Modus zugreifen können, stellen Sie sicher, dass Sie die neueste Engine-Patch-Version auf Ihrem Cluster ausführen.

Einschränkungen für Dual-Stack-Netzwerkcluster

Die folgenden Einschränkungen gelten für Dual-Stack-Netzwerkcluster:

- Cluster können das IPv6-Protokoll nicht ausschließlich verwenden. Sie können ausschließlich IPv4 oder das IPv4- und das IPv6-Protokoll (Dual-Stack-Modus) verwenden.
- Amazon DocumentDB unterstützt keine nativen IPv6-Subnetze.
- Cluster, die den Dual-Stack-Modus verwenden, müssen privat sein. Sie dürfen nicht öffentlich zugänglich sein.

Erstellen eines Clusters in einer VPC

Die folgenden Verfahren helfen Ihnen beim Erstellen eines Clusters in einer VPC. Um die Standard-VPC zu verwenden, können Sie mit Schritt 2 beginnen und die VPC und die Subnetzgruppe verwenden, die bereits für Sie erstellt wurden. Bei Bedarf können Sie auch zusätzliche VPCs erstellen.

> **Note**
>
> Wenn Ihr Cluster in der VPC öffentlich zugänglich sein soll, müssen Sie die DNS-Informationen für die VPC aktualisieren, indem Sie die VPC-Attribute und aktivieren. `DNS hostnames` `DNS resolution` Informationen zum Aktualisieren des DNS für eine VPC-Instance finden Sie unter Anzeigen und Aktualisieren von DNS-Attributen für Ihre VPC im Amazon Virtual Private Cloud-Benutzerhandbuch.

Gehen Sie wie folgt vor, um einen Cluster in einer VPC zu erstellen:

- Schritt 1: Erstellen einer VPC
- Schritt 2: Erstellung einer Subnetzgruppe
- Schritt 3: Erstellen einer VPC-Sicherheitsgruppe
- Schritt 4: Erstellen Sie einen Cluster in der VPC

Schritt 1: Erstellen einer VPC

Erstellen Sie eine VPC mit Subnetzen in mindestens zwei Availability Zones. Sie verwenden diese Subnetze, wenn Sie eine Subnetzgruppe erstellen. Wenn Sie eine Standard-VPC verwenden, wird automatisch ein Subnetz in jeder Availability Zone der AWS-Region für Sie erstellt.

Weitere Informationen finden Sie unter Erstellen Sie eine IPv4-only VPC zur Verwendung mit einem DocumentDB-Cluster oder unter Create a VPC im Amazon Virtual Private Cloud-Benutzerhandbuch.

Schritt 2: Erstellung einer Subnetzgruppe

Eine Subnetzgruppe ist eine Sammlung von Subnetzen (in der Regel privat), die Sie für eine VPC erstellen und die Sie dann für Ihre Cluster festlegen. Eine Subnetzgruppe ermöglicht es Ihnen, eine bestimmte VPC anzugeben, wenn Sie Cluster mithilfe der oder der DocumentDB-API erstellen. AWS CLI Wenn Sie die verwenden AWS-Managementkonsole, können Sie einfach die VPC und Subnetze auswählen, die Sie verwenden möchten. Jede Subnetzgruppe muss mindestens ein Subnetz in mindestens zwei Availability Zones in der haben. AWS-Region Als bewährte Methode sollte jede Subnetzgruppe mindestens ein Subnetz für jede Availability Zone in der haben. AWS-Region

Damit ein Cluster öffentlich zugänglich ist, müssen die Subnetze in der Subnetzgruppe über ein Internet-Gateway verfügen. Weitere Informationen zu Internet-Gateways für Subnetze finden Sie unter Aktivieren des Internetzugangs für eine VPC mithilfe eines Internet-Gateways im Amazon Virtual Private Cloud-Benutzerhandbuch.

> **Note**
>
> Die Subnetzgruppe für eine lokale Zone kann nur ein Subnetz haben.

Wenn Sie einen Cluster in einer VPC erstellen, können Sie eine Subnetzgruppe auswählen. Amazon DocumentDB wählt ein Subnetz und eine IP-Adresse (innerhalb dieses Subnetzes) aus, die Ihrem Cluster zugeordnet werden sollen. Wenn keine Subnetzgruppen vorhanden sind, erstellt Amazon DocumentDB beim Erstellen des Clusters eine Standard-Subnetzgruppe. DocumentDB erstellt ein Elastic Network Interface und ordnet es Ihrem Cluster mit dieser IP-Adresse zu. Der Cluster verwendet die Availability Zone, die das Subnetz enthält.

In diesem Schritt erstellen Sie eine Subnetzgruppe und fügen die Subnetze hinzu, die Sie für Ihre VPC erstellt haben.

Um eine Subnetzgruppe zu erstellen

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Wählen Sie im Navigationsbereich Subnetzgruppe aus.

3. Wählen Sie Erstellen aus.
4. Geben Sie unter Name den Namen Ihrer Subnetzgruppe ein.
5. Geben Sie unter Beschreibung eine Beschreibung für Ihre Subnetzgruppe ein.
6. Wählen Sie im Abschnitt Subnetze hinzufügen für VPC die Standard-VPC oder die von Ihnen erstellte VPC aus. Wählen Sie dann die Availability Zones aus, die die Subnetze aus Availability Zones enthalten, und wählen Sie dann die Subnetze aus Subnets aus.
7. Wählen Sie Erstellen aus.

   Ihre neue Subnetzgruppe wird in der Liste der Subnetzgruppen in der DocumentDB-Konsole angezeigt. Sie können die Subnetzgruppe auswählen, um Details, einschließlich aller Subnetze, die der Gruppe zugeordnet sind, im Detailbereich unten im Fenster anzuzeigen.

Schritt 3: Erstellen einer VPC-Sicherheitsgruppe

Bevor Sie Ihren Cluster erstellen, erstellen Sie eine VPC-Sicherheitsgruppe, die Sie ihm zuordnen möchten. Wenn Sie keine VPC-Sicherheitsgruppe erstellen, können Sie beim Erstellen des Clusters die Standardsicherheitsgruppe verwenden. Anweisungen zum Erstellen einer Sicherheitsgruppe für Ihren Cluster finden Sie unter oder unter Erstellen Sie eine IPv4-only VPC zur Verwendung mit einem DocumentDB-Cluster Steuern des Datenverkehrs zu Ihren AWS Ressourcen mithilfe von Sicherheitsgruppen im Amazon Virtual Private Cloud-Benutzerhandbuch.

Schritt 4: Erstellen Sie einen Cluster in der VPC

In diesem Schritt erstellen Sie einen Cluster mit dem VPC-Namen, der Subnetzgruppe und der VPC-Sicherheitsgruppe, die Sie in den vorherigen Schritten erstellt haben.

> **Note**
>
> Wenn Ihr Cluster in der VPC öffentlich zugänglich sein soll, müssen Sie die VPC-Attribute und aktivieren. `DNS hostnames` `DNS resolution` Weitere Informationen finden Sie unter Anzeigen und Aktualisieren von DNS-Attributen für Ihre VPC im Amazon Virtual Private Cloud-Benutzerhandbuch.

Einzelheiten zum Erstellen eines Clusters finden Sie unterErstellen eines Amazon DocumentDB-Clusters.

Wenn Sie im Abschnitt Konnektivität dazu aufgefordert werden, geben Sie den VPC-Namen, die Subnetzgruppe und die VPC-Sicherheitsgruppe ein.

> **Note**
>
> Das Aktualisieren von VPCs wird derzeit für DocumentDBDocumentDB-Cluster nicht unterstützt.

Zugreifen auf einen Amazon DocumentDB-Cluster in einer VPC

Amazon DocumentDB unterstützt die folgenden Szenarien für den Zugriff auf einen Cluster in einer VPC:

Topics

- Ein Cluster in einer VPC, auf den eine Amazon EC2-Instance in derselben VPC zugreift
- Ein Cluster in einer VPC, auf den eine Amazon EC2-Instance in einer anderen VPC zugreift

Ein Cluster in einer VPC, auf den eine Amazon EC2-Instance in derselben VPC zugreift

Ein Cluster in einer VPC wird häufig verwendet, um Daten mit einem Anwendungsserver zu teilen, der in einer Amazon EC2-Instance in derselben VPC ausgeführt wird.

Nachfolgend finden Sie den einfachsten Weg für die Verwaltung des Zugriffs zwischen EC2-Instances und DB-Instances in derselben VPC:

- Erstellen Sie eine VPC-Sicherheitsgruppe, in der sich Ihre Cluster befinden sollen. Diese Sicherheitsgruppe kann verwendet werden, um den Zugriff auf die Cluster einzuschränken. Sie können beispielsweise eine benutzerdefinierte Regel für diese Sicherheitsgruppe erstellen. Dies ermöglicht möglicherweise den TCP-Zugriff über den Port, den Sie dem Cluster bei der Erstellung zugewiesen haben, und über eine IP-Adresse, die Sie für den Zugriff auf den Cluster zu Entwicklungs- oder anderen Zwecken verwenden.
- Erstellen Sie eine VPC-Sicherheitsgruppe, der sich Ihre EC2-Instances (Webserver und Clients) befinden. Mithilfe dieser Sicherheitsgruppe können Sie bei Bedarf den Internetzugriff auf die EC2-Instance über die Routing-Tabelle der VPC zulassen. Sie können beispielsweise Regeln für diese Sicherheitsgruppe festlegen, damit der TCP-Zugriff auf die EC2-Instance über Port 22 möglich ist.

- Erstellen Sie benutzerdefinierte Regeln in der Sicherheitsgruppe für Ihre Cluster, die Verbindungen aus der Sicherheitsgruppe zulassen, die Sie für Ihre EC2-Instances erstellt haben. Diese Regeln ermöglichen möglicherweise jedem Mitglied der Sicherheitsgruppe den Zugriff auf die Cluster.

Es gibt ein zusätzliches öffentliches und privates Subnetz in einer separaten Availability Zone. Eine DocumentDB-Subnetzgruppe benötigt ein Subnetz in mindestens zwei Availability Zones. Das zusätzliche Subnetz macht es einfach, in Zukunft auf eine Multi-AZ-Cluster-Bereitstellung umzusteigen.

Anweisungen zum Erstellen einer VPC mit öffentlichen und privaten Subnetzen für dieses Szenario finden Sie unter. Erstellen Sie eine IPv4-only VPC zur Verwendung mit einem DocumentDB-Cluster

> **Tip**
>
> Sie können die Netzwerkkonnektivität zwischen einer Amazon EC2-Instance und einem DocumentDB-Cluster automatisch einrichten, wenn Sie den Cluster erstellen. Weitere Informationen finden Sie unter Stellen Sie automatisch eine Verbindung zu Amazon EC2 her.

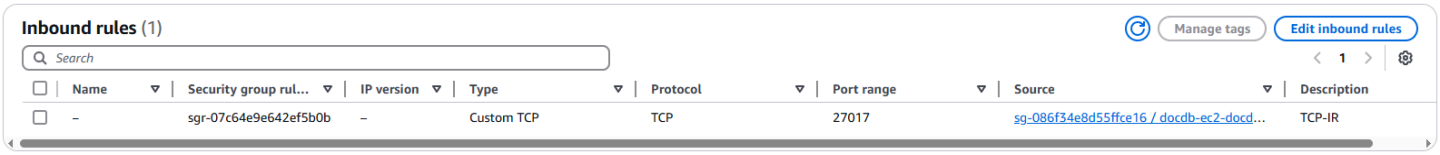
Gehen Sie wie folgt vor, um eine Regel in einer VPC-Sicherheitsgruppe zu erstellen, die Verbindungen aus einer anderen Sicherheitsgruppe zulässt:

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Amazon VPC-Konsole unter. https://console.aws.amazon.com/vpc
2. Suchen Sie im Navigationsbereich nach Sicherheitsgruppen und wählen Sie sie aus.
3. Erstellen oder wählen Sie eine Sicherheitsgruppe aus, der Sie den Zugriff zu Teilen einer anderen Sicherheitsgruppe erlauben möchten. Dies ist die Sicherheitsgruppe, die Sie für Ihre Cluster verwenden. Wählen Sie die Registerkarte Inbound Rules (Eingehende Regeln) und anschließend Edit Inbound Rules (Eingehende Regeln bearbeiten) aus.
4. Wählen Sie auf der Seite Edit inbound rules (Regeln für eingehenden Datenverkehr bearbeiten) die Option Add Rule (Regel hinzufügen).
5. Wählen Sie für Typ den Eintrag aus, der dem Port entspricht, den Sie bei der Erstellung Ihres Clusters verwendet haben, z. B. Custom TCP.
6. Beginnen Sie im Feld Quelle mit der Eingabe der ID der Sicherheitsgruppe, in der die entsprechenden Sicherheitsgruppen aufgeführt sind. Wählen Sie die Sicherheitsgruppe mit den Mitgliedern aus, die Zugriff auf die durch diese Sicherheitsgruppe geschützten Ressourcen

erhalten sollen. Im vorhergehenden Szenario ist dies die Sicherheitsgruppe, die Sie für Ihre EC2-Instance verwenden.

7. Falls erforderlich, wiederholen Sie die Schritte für das TCP-Protokoll, indem Sie eine Regel mit All TCP als Typ und Ihrer Sicherheitsgruppe im Feld Source erstellen. Wenn Sie das UDP-Protokoll verwenden möchten, erstellen Sie eine Regel mit Alle UDP als Typ und Ihrer Sicherheitsgruppe im Quelle.
8. Wählen Sie Regeln speichern aus.

Der folgende Bildschirm zeigt eine eingehende Regel mit einer Sicherheitsgruppe für ihre Quelle.



Weitere Informationen zum Herstellen einer Verbindung zu einem Cluster von Ihrer EC2-Instance aus finden Sie unter. Stellen Sie automatisch eine Verbindung zu Amazon EC2 her

Ein Cluster in einer VPC, auf den eine Amazon EC2-Instance in einer anderen VPC zugreift

Wenn sich Ihre Cluster in einer anderen VPC befinden als die EC2-Instance, mit der Sie darauf zugreifen, können Sie VPC-Peering verwenden, um auf den Cluster zuzugreifen.

Peering: Eine VPC-Peering-Verbindung ist eine Netzwerkverbindung zwischen zwei VPCs. Diese ermöglicht die Weiterleitung des Datenverkehrs zwischen den VPCs mithilfe von privaten IP-Adressen. Ressourcen in jeder der VPCs können so miteinander kommunizieren, als befänden sie sich im selben Netzwerk. Sie können eine VPC-Peering-Verbindung zwischen Ihren eigenen VPCs, mit einer VPC in einem anderen Konto oder mit einer VPC in einem anderen AWS Konto herstellen. AWS-Region Weitere Informationen über VPC-Peering finden Sie unter VPC-Peering im Amazon Virtual Private Cloud-Benutzerhandbuch.

Erstellen Sie eine IPv4-only VPC zur Verwendung mit einem DocumentDB-Cluster

Ein gängiges Szenario beinhaltet einen Cluster in einer Virtual Private Cloud (VPC), der auf dem Amazon VPC-Service basiert. Diese VPC könnte beispielsweise Daten mit einem Dienst oder einer Anwendung teilen, die in derselben VPC ausgeführt wird. In diesem Thema erstellen Sie die VPC für dieses Szenario.

Topics

- Schritt 1: Erstellen Sie eine VPC mit privaten und öffentlichen Subnetzen
- Schritt 2: Erstellen Sie eine VPC-Sicherheitsgruppe für eine öffentliche Anwendung
- Schritt 3: Erstellen Sie eine VPC-Sicherheitsgruppe für einen privaten Cluster
- Schritt 4: Erstellen Sie eine Subnetzgruppe
- Löschen einer VPC

Ihr Cluster muss nur für Ihre Anwendung und nicht für das öffentliche Internet verfügbar sein. Daher erstellen Sie eine VPC sowohl mit öffentlichen als auch mit privaten Subnetzen. Die Anwendung wird im öffentlichen Subnetz gehostet, sodass sie das öffentliche Internet erreichen kann. Der Cluster wird in einem privaten Subnetz gehostet. Die Anwendung kann eine Verbindung zum Cluster herstellen, da sie in derselben VPC gehostet wird. Der Cluster ist jedoch nicht für das öffentliche Internet verfügbar, was eine höhere Sicherheit bietet.

Das Verfahren in diesem Thema konfiguriert ein zusätzliches öffentliches und privates Subnetz in einer separaten Availability Zone. Diese Subnetze werden von dem Verfahren nicht verwendet. Eine DocumentDB-Subnetzgruppe benötigt ein Subnetz in mindestens zwei Availability Zones. Das zusätzliche Subnetz erleichtert die Konfiguration von mehr als einer DocumentDB-Instance.

In diesem Thema wird die Konfiguration einer VPC für Amazon DocumentDB-Cluster beschrieben. Weitere Informationen zur Amazon VPC-Sicherheit finden Sie unter Sicherheit im Amazon VPC-Benutzerhandbuch.

> **Tip**
>
> Sie können die Netzwerkkonnektivität zwischen einer Amazon EC2-Instance und einem DocumentDB-Cluster automatisch einrichten, wenn Sie den Cluster erstellen. Die Netzwerkkonfiguration ähnelt der in diesem Szenario beschriebenen. Weitere Informationen finden Sie unter Stellen Sie automatisch eine Verbindung zu Amazon EC2 her.

Schritt 1: Erstellen Sie eine VPC mit privaten und öffentlichen Subnetzen

Verwenden sie die folgenden Vorgänge, um eine VPC sowohl mit öffentlichen als auch mit privaten Subnetzen zu erstellen.

So erstellen Sie eine VPC und Subnetze

1. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc.
2. Wählen Sie in der oberen rechten Ecke von die Region aus AWS-Managementkonsole, in der Sie Ihre VPC erstellen möchten. In diesem Beispiel wird die Region USA West (Oregon) verwendet.
3. Wählen Sie links oben die Option VPC-Dashboard aus. Wählen Sie Create VPC (VPC erstellen) aus, um mit dem Erstellen einer VPC zu beginnen.
4. Wählen Sie unter VPC Settings (VPC-Einstellungen) für Resources to create (Zu erstellende Ressourcen) VPC and more (VPC und mehr) aus.
5. Legen Sie für VPC settings (VPC-Einstellungen) folgende Werte fest:
   - Automatische Generierung von Namensschildern — **`example`**
   - IPv4-CIDR-Block — **`10.0.0.0/16`**
   - IPv6-CIDR-Block — Kein IPv6-CIDR-Block
   - Tenancy — Standard
   - Anzahl der Availability Zones (AZs) — 2
   - AZs anpassen — Behalten Sie die Standardwerte bei
   - Anzahl der öffentlichen Subnetze — 2
   - Anzahl der privaten Subnetze — 2
   - Passen Sie die CIDR-Blöcke der Subnetze an — Behalten Sie die Standardwerte bei
   - NAT-Gateways ($) — Keine
   - VPC-Endpunkte — Keine
   - DNS-Optionen — Behalten Sie die Standardwerte bei
6. Wählen Sie VPC erstellen aus.

Schritt 2: Erstellen Sie eine VPC-Sicherheitsgruppe für eine öffentliche Anwendung

Erstellen Sie als Nächstes eine Sicherheitsgruppe für den öffentlichen Zugriff. Wenn Sie eine Verbindung mit öffentlichen EC2-Instances in Ihrer VPC herstellen möchten, fügen Sie Ihrer VPC-Sicherheitsgruppe Regeln für eingehenden Datenverkehr hinzu. Diese ermöglichen die Verbindung des Datenverkehrs aus dem Internet.

Um eine VPC-Sicherheitsgruppe zu erstellen

1. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc.

2. Wählen Sie VPC Dashboard (VPC-Dashboard), Security Groups (Sicherheitsgruppen) und anschließend Create security group (Sicherheitsgruppe erstellen).
3. Legen Sie auf der Seite Create security group (Sicherheitsgruppe erstellen) die folgenden Werte fest:

   - Name der Sicherheitsgruppe — **`example-securitygroup`**
   - Beschreibung — **`Application security group`**
   - VPC — Wählen Sie die VPC, die Sie zuvor erstellt haben, zum Beispiel: vpc-example.
4. Fügen Sie der Sicherheitsgruppe Regeln für den eingehenden Datenverkehr hinzu.

   a. Legen Sie die IP-Adresse fest, die für die Verbindung mit EC2-Instances in Ihrer VPC mit Secure Shell (SSH) verwendet werden soll. Um Ihre öffentliche IP-Adresse in einem anderen Browserfenster oder einer anderen Registerkarte zu ermitteln, können Sie den Dienst unter verwenden. https://checkip.amazonaws.com Ein Beispiel für eine IP-Adresse ist `203.0.113.25/32`.

   In vielen Fällen können Sie eine Verbindung über einen Internetdienstanbieter (ISP) oder hinter Ihrer Firewall ohne statische IP-Adresse herstellen. Suchen Sie in diesem Fall den Bereich der IP-Adressen, die von Client-Computern verwendet werden.

   > ⚠ Warning
   >
   > Wenn Sie `0.0.0.0/0` für SSH-Zugriff verwenden, ermöglichen Sie für alle IP-Adressen den Zugriff auf Ihre öffentlichen Instances. Dieser Ansatz ist zwar für kurze Zeit in einer Testumgebung zulässig, aber für Produktionsumgebungen sehr unsicher. Für die Produktion wird nur eine bestimmte IP-Adresse bzw. ein bestimmter Adressbereich für den Zugriff auf Ihre Instances autorisiert.

   b. Wählen Sie im Abschnitt Eingehende Regeln die Option Regel hinzufügen aus.

   c. Legen Sie die folgenden Werte für Ihre neue eingehende Regel fest, um SSH den Zugriff auf Ihre Amazon-EC2-Instance zu erlauben. Nachdem Sie dies getan haben, können Sie eine Verbindung zu Ihrer EC2-Instance herstellen, um die Anwendung und andere Dienstprogramme zu installieren. Sie stellen auch eine Verbindung zu Ihrer EC2-Instance her, um Inhalte für Ihre Anwendung hochzuladen.

      - Geben Sie — ein **`SSH`**

- Quelle — Die IP-Adresse oder der Bereich, den Sie in Schritt a erstellt haben, zum Beispiel: **`203.0.113.25/32`**

d. Wählen Sie Regel hinzufügen aus.

e. Legen Sie die folgenden Werte für Ihre neue Regel für eingehenden Datenverkehr fest, um den HTTP-Zugriff auf Ihre Anwendung zu ermöglichen:

- Geben Sie — ein **`HTTP`**
- Quelle — **`0.0.0.0/0`**

5. Wählen Sie Create security group (Sicherheitsgruppe erstellen) aus, um die Sicherheitsgruppe zu erstellen.

   Notieren Sie sich die Sicherheitsgruppen-ID, da Sie sie später in einem anderen Verfahren benötigen.

Schritt 3: Erstellen Sie eine VPC-Sicherheitsgruppe für einen privaten Cluster

Um Ihren Cluster privat zu halten, erstellen Sie eine zweite Sicherheitsgruppe für den privaten Zugriff. Um eine Verbindung zu privaten Clustern in Ihrer VPC herzustellen, fügen Sie Ihrer VPC-Sicherheitsgruppe Regeln für eingehenden Datenverkehr hinzu, die nur den Datenverkehr von Ihrer Anwendung zulassen.

Um eine VPC-Sicherheitsgruppe zu erstellen

1. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc.
2. Wählen Sie VPC Dashboard (VPC-Dashboard), Security Groups (Sicherheitsgruppen) und anschließend Create security group (Sicherheitsgruppe erstellen).
3. Legen Sie auf der Seite Create security group (Sicherheitsgruppe erstellen) die folgenden Werte fest:

   - Name der Sicherheitsgruppe — **`example-securitygroup`**
   - Beschreibung — **`Instance security group`**
   - VPC — Wählen Sie die VPC, die Sie zuvor erstellt haben, zum Beispiel: vpc-example

4. Fügen Sie der Sicherheitsgruppe Regeln für den eingehenden Datenverkehr hinzu.

   a. Wählen Sie im Abschnitt Eingehende Regeln die Option Regel hinzufügen aus.

b. Legen Sie die folgenden Werte für Ihre neue Regel für eingehenden Datenverkehr fest, um DocumentDB-Verkehr auf Port 27017 von Ihrer Amazon EC2-Instance aus zuzulassen. Nachdem Sie dies getan haben, können Sie von Ihrer Anwendung aus eine Verbindung zu Ihrem Cluster herstellen. Auf diese Weise können Sie Daten aus Ihrer Anwendung speichern und in Ihrer Datenbank abrufen.

   - Geben Sie — ein **`Custom TCP`**
   - Quelle — Die ID der Anwendungssicherheitsgruppe, die Sie zuvor in diesem Thema erstellt haben, zum Beispiel: sg-9edd5cfb.

c. Wählen Sie Regel hinzufügen aus.

d. Legen Sie die folgenden Werte für Ihre neue Regel für eingehenden Datenverkehr fest, um den HTTP-Zugriff auf Ihre Anwendung zuzulassen:

   - Geben Sie — ein **`HTTP`**
   - Quelle — **`0.0.0.0/0`**

5. Wählen Sie Create security group (Sicherheitsgruppe erstellen) aus, um die Sicherheitsgruppe zu erstellen.

Schritt 4: Erstellen Sie eine Subnetzgruppe

Eine Subnetzgruppe ist eine Sammlung von Subnetzen, die Sie in einer VPC erstellen und die Sie dann für Ihre Cluster festlegen. Eine Subnetzgruppe ermöglicht es Ihnen, beim Erstellen von Clustern eine bestimmte VPC anzugeben.

Um eine Subnetzgruppe zu erstellen

1. Identifizieren Sie die privaten Subnetze für Ihre Datenbank in der VPC.

   a. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc.

   b. Wählen Sie VPC Dashboard (VPC-Dashboard) und dann Subnets (Subnetze) aus.

   c. Notieren Sie sich die Subnetz-IDs der Subnetze, die Sie in Schritt 1 erstellt haben, zum Beispiel: example-subnet-private1-us-west-2a und example-subnet-private2-us-west-2b. Sie benötigen die Subnetz-IDs, wenn Sie Ihre Subnetzgruppe erstellen.

2. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb

Stellen Sie sicher, dass Sie eine Verbindung zur Amazon DocumentDB-Konsole herstellen, nicht zur Amazon VPC-Konsole.

3. Wählen Sie im Navigationsbereich Subnetzgruppe aus.
4. Wählen Sie Erstellen aus.
5. Legen Sie auf der Seite „Subnetzgruppe erstellen“ im Abschnitt „Subnetzgruppendetails“ die folgenden Werte fest:
   - Name — **`example-db-subnet-group`**
   - Beschreibung — **`Instance security group`**
6. Legen Sie im Abschnitt Subnetze hinzufügen die folgenden Werte fest:
   - VPC — Wählen Sie die VPC, die Sie zuvor erstellt haben, zum Beispiel: vpc-example
   - Availability Zones — Wählen Sie beide Availability Zones aus, die in Schritt 1 erstellt wurden. Beispiel: us-west-2a und us-west-2b
   - Subnetze — Wählen Sie die privaten Subnetze aus, die Sie in Schritt 1 erstellt haben.
7. Wählen Sie Erstellen aus.

Ihre neue Subnetzgruppe wird in der Liste der Subnetzgruppen auf der DocumentDB-Konsole angezeigt. Sie können die Subnetzgruppe auswählen, um Details im Detailbereich anzuzeigen. Diese Informationen umfassen alle Subnetze, die der Gruppe zugeordnet sind.

> **Note**
>
> Wenn Sie diese VPC erstellt haben, um sie einem DocumentDB-Cluster zuzuordnen, erstellen Sie den Cluster, indem Sie den Anweisungen unter folgen. Erstellen eines Amazon DocumentDB-Clusters

Löschen einer VPC

Sie können eine VPC und die anderen darin verwendeten Ressourcen löschen, wenn sie nicht mehr benötigt werden.

> **Note**
>
> Wenn Sie der VPC, die Sie in diesem Thema erstellt haben, Ressourcen hinzugefügt haben, müssen Sie diese möglicherweise löschen, bevor Sie die VPC löschen können. Zu diesen Ressourcen können beispielsweise Amazon EC2-Instances oder DocumentDB-Cluster gehören. Weitere Informationen finden Sie unter Löschen Ihrer VPC im Amazon VPC User Guide.

Um eine VPC und zugehörige Ressourcen zu löschen

1. Löschen Sie die Subnetzgruppe:
   a. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
   b. Wählen Sie im Navigationsbereich Subnetzgruppe aus.
   c. Wählen Sie die Subnetzgruppe aus, die Sie löschen möchten, z. B. example-db-subnet-group.
   d. Wählen Sie Löschen, und wählen Sie dann im Bestätigungsfenster Löschen.
2. Notieren Sie sich die VPC-ID.
   a. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc.
   b. Wählen Sie VPC Dashboard und dann Your VPCs aus.
   c. Identifizieren Sie in der Liste die VPC, die Sie erstellt haben, z. B. vpc-example.
   d. Notieren Sie die VPC ID (VPC-ID) der VPC, die Sie erstellt haben. Sie benötigen die VPC-ID in späteren Schritten.
3. Löschen der Sicherheitsgruppen:
   a. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc.
   b. Wählen Sie VPC Dashboard und dann Sicherheitsgruppen.
   c. Wählen Sie die Sicherheitsgruppe für den Amazon DocumentDB-Cluster aus, z. B. example-securitygroup.
   d. Wählen Sie für Aktionen die Option Sicherheitsgruppen löschen und dann im Bestätigungsdialogfeld Löschen aus.
   e. Zurück auf der Seite „Sicherheitsgruppen“ wählen Sie die Sicherheitsgruppe für die Amazon EC2-Instance aus, z. B. example-securitygroup.

f. Wählen Sie für Aktionen die Option Sicherheitsgruppen löschen und dann im Bestätigungsdialogfeld Löschen aus.

4. Löschen der VPC:

a. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc.

b. Wählen Sie VPC-Dashboard und anschließend Ihre VPCs aus.

c. Wählen Sie die VPC aus, die Sie löschen möchten, z. B. vpc-example.

d. Wählen Sie für Aktionen die Option VPC löschen aus.

Auf der Bestätigungsseite werden weitere Ressourcen angezeigt, die der VPC zugeordnet sind, die ebenfalls gelöscht werden, einschließlich der damit verknüpften Subnetze.

e. Geben Sie im Bestätigungsdialogfeld den Text ein**`delete`**, und wählen Sie dann Löschen aus.

Erstellen Sie eine Dual-Stack-VPC zur Verwendung mit einem DocumentDB-Cluster

Ein gängiges Szenario umfasst einen Cluster in einer Virtual Private Cloud (VPC), der auf dem Amazon VPC-Service basiert. Diese VPC teilt Daten mit einer öffentlichen Amazon-EC2-Instance, die in derselben VPC ausgeführt wird. In diesem Thema erstellen Sie die VPC für dieses Szenario.

Topics

- Schritt 1: Erstellen Sie eine VPC mit privaten und öffentlichen Subnetzen
- Schritt 2: Erstellen Sie eine VPC-Sicherheitsgruppe für eine öffentliche Amazon EC2-Instance
- Schritt 3: Erstellen Sie eine VPC-Sicherheitsgruppe für einen privaten Cluster
- Schritt 4: Erstellen Sie eine Subnetzgruppe
- Schritt 5: Erstellen Sie eine Amazon EC2-Instance im Dual-Stack-Modus
- Schritt 6: Erstellen Sie einen Cluster im Dual-Stack-Modus
- Schritt 7: Stellen Sie eine Verbindung zu Ihrer Amazon EC2-Instance und Ihrem DB-Cluster her
- Löschen der VPC

In diesem Verfahren erstellen Sie die VPC für dieses Szenario, die mit einer Datenbank arbeitet, die im Dual-Stack-Modus läuft. Dual-stack Der Modus ermöglicht die Verbindung über das IPv6-

Adressierungsprotokoll. Weitere Informationen über die IP-Adressierung finden Sie unter Amazon DocumentDB-IP-Adressierung.

Dual-stack Netzwerkcluster werden in den meisten Regionen unterstützt. Weitere Informationen finden Sie unter Dual-stack Modus Regions- und Versionsverfügbarkeit. Informationen zu den Einschränkungen des Dual-Stack-Modus finden Sie unter Einschränkungen für Dual-Stack-Netzwerkcluster.

Dieses Thema und das IPv4-only Thema erstellen die öffentlichen und privaten Subnetze in derselben VPC. Informationen zum Erstellen des Amazon DocumentDB-Clusters in einer VPC und der Amazon EC2-Instance in einer anderen VPC finden Sie unter. Zugreifen auf einen Amazon DocumentDB-Cluster in einer VPC

Ihr DocumentDB-Cluster muss nur für Ihre Amazon EC2-Instance und nicht für das öffentliche Internet verfügbar sein. Daher erstellen Sie eine VPC sowohl mit öffentlichen als auch mit privaten Subnetzen. Die EC2-Instance wird im öffentlichen Subnetz gehostet, sodass sie das öffentliche Internet erreichen kann. Der Cluster wird in einem privaten Subnetz gehostet. Die EC2-Instance kann eine Verbindung zum Cluster herstellen, da sie in derselben VPC gehostet wird. Der Cluster ist jedoch nicht für das öffentliche Internet verfügbar, was eine höhere Sicherheit bietet.

Die Verfahren in diesem Thema konfigurieren ein zusätzliches öffentliches und privates Subnetz in einer separaten Availability Zone. Diese Subnetze werden von dem Verfahren nicht verwendet. Eine DocumentDB-Subnetzgruppe benötigt ein Subnetz in mindestens zwei Availability Zones. Das zusätzliche Subnetz macht es einfach, mehr als eine DocumentDB-Instance zu konfigurieren.

Um einen Cluster zu erstellen, der den Dual-Stack-Modus verwendet, geben Sie Dual-stack mode für die Einstellung Netzwerktyp an. Sie können auch einen Cluster mit derselben Einstellung ändern. Weitere Hinweise zum Erstellen eines Clusters finden Sie unterErstellen eines Amazon DocumentDB-Clusters. Weitere Informationen über das Ändern eines DB-Clusters finden Sie unter Ändern eines Amazon DocumentDB-Clusters.

Dieses Thema beschreibt die Konfiguration einer VPC für Amazon DocumentDB-Cluster. Weitere Informationen zur Amazon VPC-Sicherheit finden Sie unter Sicherheit im Amazon VPC-Benutzerhandbuch.

Schritt 1: Erstellen Sie eine VPC mit privaten und öffentlichen Subnetzen

Verwenden sie die folgenden Vorgänge, um eine VPC sowohl mit öffentlichen als auch mit privaten Subnetzen zu erstellen.

So erstellen Sie eine VPC und Subnetze

1. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc.
2. Wählen Sie in der oberen rechten Ecke von die Region aus AWS-Managementkonsole, in der Sie Ihre VPC erstellen möchten. In diesem Beispiel wird die Region USA West (Oregon) verwendet.
3. Wählen Sie links oben die Option VPC-Dashboard aus. Wählen Sie Create VPC (VPC erstellen) aus, um mit dem Erstellen einer VPC zu beginnen.
4. Wählen Sie unter VPC Settings (VPC-Einstellungen) für Resources to create (Zu erstellende Ressourcen) VPC and more (VPC und mehr) aus.
5. Legen Sie für VPC settings (VPC-Einstellungen) folgende Werte fest:
   - Automatische Generierung von Namensschildern — **`example-dual-stack`**
   - IPv4-CIDR-Block — **`10.0.0.0/16`**
   - IPv6-CIDR-Block — IPv6-CIDR-Block Amazon-provided
   - Tenancy — Standard
   - Anzahl der Availability Zones (AZs) — 2
   - AZs anpassen — Behalten Sie die Standardwerte bei
   - Anzahl der öffentlichen Subnetze — 2
   - Anzahl der privaten Subnetze — 2
   - Passen Sie die CIDR-Blöcke der Subnetze an — Behalten Sie die Standardwerte bei
   - NAT-Gateways ($) — Keine
   - Internet-Gateway nur für ausgehenden Verkehr — Nein
   - VPC-Endpunkte — Keine
   - DNS-Optionen — Behalten Sie die Standardwerte bei
6. Wählen Sie VPC erstellen aus.

Schritt 2: Erstellen Sie eine VPC-Sicherheitsgruppe für eine öffentliche Amazon EC2-Instance

Als Nächstes erstellen Sie eine Sicherheitsgruppe für öffentlichen Zugriff. Wenn Sie eine Verbindung mit öffentlichen EC2-Instances in Ihrer VPC herstellen möchten, fügen Sie Ihrer VPC-Sicherheitsgruppe eingehende Regeln hinzu, die Verbindungen aus dem Internet zulassen.

Um eine VPC-Sicherheitsgruppe zu erstellen

1. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc.
2. Wählen Sie VPC Dashboard (VPC-Dashboard), Security Groups (Sicherheitsgruppen) und anschließend Create security group (Sicherheitsgruppe erstellen).
3. Legen Sie auf der Seite Create security group (Sicherheitsgruppe erstellen) die folgenden Werte fest:

   - Name der Sicherheitsgruppe — **`example-dual-stack-securitygroup`**
   - Beschreibung — **`Dual-stack security group`**
   - VPC — Wählen Sie die VPC, die Sie zuvor erstellt haben, zum Beispiel: vpc-example-dual-stack.
4. Fügen Sie der Sicherheitsgruppe Regeln für den eingehenden Datenverkehr hinzu.

   a. Legen Sie die IP-Adresse fest, die für die Verbindung mit EC2-Instances in Ihrer VPC mit Secure Shell (SSH) verwendet werden soll. Um Ihre öffentliche IP-Adresse in einem anderen Browserfenster oder einer anderen Registerkarte zu ermitteln, können Sie den Dienst unter verwenden. https://checkip.amazonaws.com

      Ein Beispiel für einen IPv4-Adressbereich (Internet Protocol Version 4) ist`203.0.113.25/32`. Ein Beispiel für einen Internet Protocol Version 6 (IPv6)-Adressbereich ist `2001:db8:1234:1a00::/64`.

      In vielen Fällen können Sie eine Verbindung über einen Internetdienstanbieter (ISP) oder hinter Ihrer Firewall ohne statische IP-Adresse herstellen. Suchen Sie in diesem Fall den Bereich der IP-Adressen, die von Client-Computern verwendet werden.

      > **Warning**
      >
      > Wenn Sie `0.0.0.0/0` für IPv4 oder `::0` für IPv6 verwenden, lassen Sie für alle IP-Adressen den Zugriff auf Ihre öffentlichen Instances mit SSH zu. Dieser Ansatz ist zwar für kurze Zeit in einer Testumgebung zulässig, aber für Produktionsumgebungen sehr unsicher. Autorisieren Sie in Produktionsumgebungen nur eine bestimmte IP-Adresse bzw. einen bestimmten Adressbereich für den Zugriff auf Ihre Instances.

   b. Wählen Sie im Abschnitt Eingehende Regeln die Option Regel hinzufügen aus.

   c. Legen Sie die folgenden Werte für Ihre neue eingehende Regel fest, um SSH den Zugriff auf Ihre Amazon-EC2-Instance zu erlauben. Nachdem Sie dies getan haben, können

Sie eine Verbindung zu Ihrer EC2-Instance herstellen, um Anwendungen oder andere Dienstprogramme zu installieren. Geben Sie eine IP-Adresse an, damit Sie auf Ihre EC2-Instance zugreifen können:

- Geben Sie — ein **`SSH`**
- Quelle — Die IP-Adresse oder der Bereich, den Sie in Schritt a erstellt haben. Ein Beispiel für einen IPv4-Adressbereich ist **`203.0.113.25/3`** 2. Ein Beispiel für einen IPv6-Adressbereich ist. **`2001:DB8::/32`**

d. Wählen Sie Regel hinzufügen aus.

5. Wählen Sie Create security group (Sicherheitsgruppe erstellen) aus, um die Sicherheitsgruppe zu erstellen.

   Notieren Sie sich die Sicherheitsgruppen-ID, da Sie sie später in einem anderen Verfahren benötigen.

Schritt 3: Erstellen Sie eine VPC-Sicherheitsgruppe für einen privaten Cluster

Um Ihren Cluster privat zu halten, erstellen Sie eine zweite Sicherheitsgruppe für den privaten Zugriff. Um eine Verbindung zu privaten Clustern in Ihrer VPC herzustellen, fügen Sie Ihrer VPC-Sicherheitsgruppe Regeln für eingehenden Verkehr hinzu. Diese lassen ausschließlich Datenverkehr von Ihrer Amazon-EC2-Instance zu.

Um eine VPC-Sicherheitsgruppe zu erstellen

1. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc.
2. Wählen Sie VPC Dashboard (VPC-Dashboard), Security Groups (Sicherheitsgruppen) und anschließend Create security group (Sicherheitsgruppe erstellen).
3. Legen Sie auf der Seite Create security group (Sicherheitsgruppe erstellen) die folgenden Werte fest:

   - Name der Sicherheitsgruppe — **`example-dual-stack-cluster-securitygroup`**
   - Beschreibung — **`Dual-stack cluster security group`**
   - VPC — Wählen Sie die VPC, die Sie zuvor erstellt haben, zum Beispiel: vpc-example-dual-stack

4. Fügen Sie der Sicherheitsgruppe Regeln für den eingehenden Datenverkehr hinzu.

   a. Wählen Sie im Abschnitt Eingehende Regeln die Option Regel hinzufügen aus.

b. Legen Sie die folgenden Werte für Ihre neue Regel für eingehenden Datenverkehr fest, um DocumentDB-Verkehr auf Port 27017 von Ihrer Amazon EC2-Instance aus zuzulassen. Nachdem Sie dies getan haben, können Sie von Ihrer EC2-Instance aus eine Verbindung zu Ihrem Cluster herstellen. Auf diese Weise können Sie Daten von Ihrer EC2-Instance an Ihre Datenbank senden.

   - Geben Sie — ein **`Custom TCP`**
   - Quelle — Die ID der EC2-Sicherheitsgruppe, die Sie zuvor in diesem Thema erstellt haben, zum Beispiel: sg-9edd5cfb.

5. Wählen Sie Create security group (Sicherheitsgruppe erstellen) aus, um die Sicherheitsgruppe zu erstellen.

Schritt 4: Erstellen Sie eine Subnetzgruppe

Eine Subnetzgruppe ist eine Sammlung von Subnetzen, die Sie in einer VPC erstellen und die Sie dann für Ihre Cluster festlegen. Mithilfe einer Subnetzgruppe können Sie beim Erstellen von Clustern eine bestimmte VPC angeben. Um eine Subnetzgruppe zu erstellen, die `DUAL` kompatibel ist, müssen alle Subnetze kompatibel sein. `DUAL` Um mit `DUAL` kompatibel zu sein, muss einem Subnetz ein IPv6 CIDR zugeordnet sein.

Um eine Subnetzgruppe zu erstellen

1. Identifizieren Sie die privaten Subnetze für Ihre Datenbank in der VPC.

   a. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc.
   b. Wählen Sie VPC Dashboard (VPC-Dashboard) und dann Subnets (Subnetze) aus.
   c. Notieren Sie sich die Subnetz-IDs der Subnetze, die Sie in Schritt 1 erstellt haben, zum Beispiel: example-dual-stack-subnet-private1-us-west-2a und example-dual-stack-subnet-private2-us-west-2b. Sie benötigen die Subnetz-IDs, wenn Sie Ihre Subnetzgruppe erstellen.

2. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb

   Stellen Sie sicher, dass Sie eine Verbindung zur Amazon DocumentDB-Konsole herstellen, nicht zur Amazon VPC-Konsole.

3. Wählen Sie im Navigationsbereich Subnetzgruppe aus.
4. Wählen Sie Erstellen aus.

5. Legen Sie auf der Seite „Subnetzgruppe erstellen“ im Abschnitt „Subnetzgruppendetails“ die folgenden Werte fest:
   - Name — **`example-dual-stack-cluster-subnet-group`**
   - Beschreibung — **`Dual-stack cluster subnet group`**
6. Stellen Sie im Abschnitt Subnetze hinzufügen die folgenden Werte ein:
   - VPC — Wählen Sie die VPC aus, die Sie zuvor erstellt haben, zum Beispiel: vpc-example-dual-stack
   - Availability Zones — Wählen Sie beide Availability Zones aus, die in Schritt 1 erstellt wurden. Beispiel: us-west-2a und us-west-2b
   - Subnetze — Wählen Sie die privaten Subnetze aus, die Sie in Schritt 1 erstellt haben.
7. Wählen Sie Erstellen aus.

Ihre neue Subnetzgruppe wird in der Liste der Subnetzgruppen auf der DocumentDB-Konsole angezeigt. Sie können die Subnetzgruppe auswählen, um Details im Detailbereich anzuzeigen. Diese Informationen umfassen alle Subnetze, die der Gruppe zugeordnet sind.

Schritt 5: Erstellen Sie eine Amazon EC2-Instance im Dual-Stack-Modus

Um eine Amazon EC2-Instance zu erstellen, folgen Sie den Anweisungen unter Starten einer EC2-Instance mithilfe des Assistenten zum Starten von Instances in der Konsole im Amazon Elastic Compute Cloud-Benutzerhandbuch.

Legen Sie auf der Seite Configure Instance Details (Konfigurieren von Instance-Detail)s wie im Folgenden gezeigt die folgenden Werte fest und behalten Sie die Standardwerte für die anderen Werte bei:

- Netzwerk — Wählen Sie eine vorhandene VPC mit öffentlichen und privaten Subnetzen aus, z. B. vpc-example-dual-stack-vpc (vpc-identifier), das in erstellt wurde. Schritt 1: Erstellen Sie eine VPC mit privaten und öffentlichen Subnetzen
- Subnetz — Wählen Sie ein vorhandenes öffentliches Subnetz aus, z. B. Subnet Identifier | example-dual-stack-subnet-public1-us-east-2a | us-east-2a, erstellt in. Schritt 2: Erstellen Sie eine VPC-Sicherheitsgruppe für eine öffentliche Amazon EC2-Instance
- Auto-assign Öffentliche IP — Wählen Sie Aktivieren.
- Auto-assign IPv6-IP — Wählen Sie „Aktivieren“.
- Firewall (Sicherheitsgruppen) Wählen Sie eine vorhandene Sicherheitsgruppe aus. —

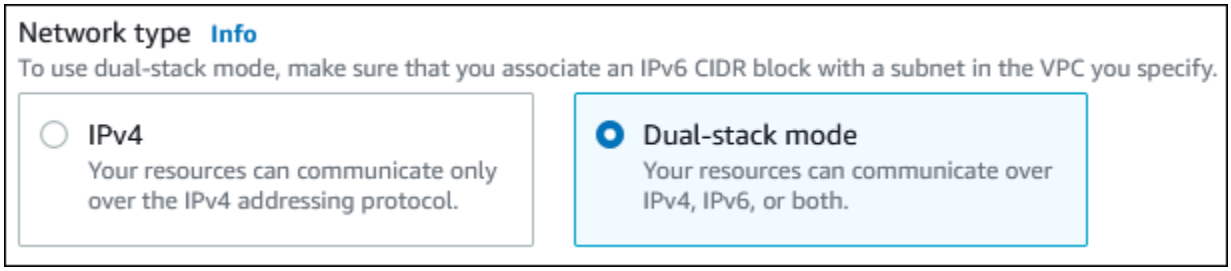
- Allgemeine Sicherheitsgruppen — Wählen Sie eine vorhandene Sicherheitsgruppe aus, z. B. die example-dual-stack-securitygroup, die in erstellt wurde. Schritt 2: Erstellen Sie eine VPC-Sicherheitsgruppe für eine öffentliche Amazon EC2-Instance Stellen Sie sicher, dass die von Ihnen gewählte Sicherheitsgruppe Regeln für eingehenden Datenverkehr für Secure Shell (SSH) und HTTP-Zugriff enthält.

Schritt 6: Erstellen Sie einen Cluster im Dual-Stack-Modus

In diesem Schritt erstellen Sie einen DB-Cluster, der im Dual-Stack-Modus ausgeführt wird. !!! Hinweis: Dieser Abschnitt muss bearbeitet werden, sobald die IPv6-Updates an der Konsole vorgenommen wurden!!!

So erstellen Sie einen Cluster im Dual-Stack-Modus

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Wählen Sie in der oberen rechten Ecke der Konsole den Ort aus, an AWS-Region dem Sie den DocumentDB-Cluster erstellen möchten. In diesem Beispiel wird die Region USA Ost (Ohio) verwendet.
3. Klicken Sie im Navigationsbereich auf Cluster.
4. Wählen Sie auf der Seite mit der Cluster-Liste die Option Create aus.
5. Stellen Sie auf der Seite „Amazon DocumentDB-Cluster erstellen“ sicher, dass die Instance-based Cluster-Option ausgewählt ist.
6. Wählen Sie im Abschnitt Konnektivität unter Netzwerktyp die Option Dual-stack Modus aus.



7. Aktivieren Sie unten auf der Seite die Option Erweiterte Einstellungen anzeigen.
8. Stellen Sie im Abschnitt Netzwerkeinstellungen die folgenden Werte ein:

   - Virtual Private Cloud (VPC) — Wählen Sie eine vorhandene VPC mit öffentlichen und privaten Subnetzen aus, z. B. vpc-example-dual-stack (vpc-identifier), das in erstellt wurde. Schritt 1: Erstellen Sie eine VPC mit privaten und öffentlichen Subnetzen

Die VPC muss Subnetze in verschiedenen Availability Zones haben.

- Subnetzgruppe — Wählen Sie eine Subnetzgruppe für die VPC aus, z. B. example-dual-stack-cluster-subnet-group created in. Schritt 4: Erstellen Sie eine Subnetzgruppe
- Öffentlicher Zugriff — Wählen Sie Nein.
- VPC-Sicherheitsgruppe (Firewall) — Wählen Sie Bestehende auswählen aus.
- Bestehende VPC-Sicherheitsgruppen — Wählen Sie eine vorhandene VPC-Sicherheitsgruppe aus, die für den privaten Zugriff konfiguriert ist, z. B. example-dual-stack-cluster-securitygroup created in. Schritt 3: Erstellen Sie eine VPC-Sicherheitsgruppe für einen privaten Cluster

  Entfernen Sie andere Sicherheitsgruppen wie die Standardsicherheitsgruppe, indem Sie das jeweilige X wählen.
- Availability Zone — Wählen Sie die Availability Zone aus, die Sie in Schritt 1 erstellt haben. Beispiel: us-west-2a.

  Um Cross-AZ-Verkehr zu vermeiden, stellen Sie sicher, dass sich der Cluster und die EC2-Instance in derselben Availability Zone befinden.

9. Geben Sie für die verbleibenden Abschnitte Ihre Cluster-Einstellungen an. Weitere Informationen zu den einzelnen Einstellungen finden Sie unter Erstellen eines Amazon DocumentDB-Clusters.

Schritt 7: Stellen Sie eine Verbindung zu Ihrer Amazon EC2-Instance und Ihrem DB-Cluster her

Nachdem Sie Ihre Amazon EC2-Instance und Ihren DocumentDB-Cluster im Dual-Stack-Modus erstellt haben, können Sie mithilfe des IPv6-Protokolls eine Verbindung zu den einzelnen Instanzen herstellen. Um mithilfe des IPv6-Protokolls eine Verbindung zu einer EC2-Instance herzustellen, folgen Sie den Anweisungen unter Verbinden Sie Ihre EC2-Instance im Amazon Elastic Compute Cloud-Benutzerhandbuch.

Um von der EC2-Instance aus eine Verbindung zu Ihrem DocumentDB-Cluster herzustellenSchritt 5: Installieren Sie die MongoDB Shell, folgen Sie den Anweisungen im Thema Amazon EC2 manuell verbinden (und fahren Sie mit den nachfolgenden Schritten 6 und Schritt 7 im gleichen Verfahren fort).

Löschen der VPC

Sie können eine VPC und die anderen darin verwendeten Ressourcen löschen, wenn sie nicht mehr benötigt werden.

> **Note**
>
> Wenn Sie der VPC, die Sie in diesem Thema erstellt haben, Ressourcen hinzugefügt haben, müssen Sie diese möglicherweise löschen, bevor Sie die VPC löschen können. Zu diesen Ressourcen können beispielsweise Amazon EC2-Instances oder DocumentDB-Cluster gehören. Weitere Informationen finden Sie unter Löschen Ihrer VPC im Amazon VPC User Guide.

Um eine VPC und zugehörige Ressourcen zu löschen

1. Löschen Sie die Subnetzgruppe:
   a. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
   b. Wählen Sie im Navigationsbereich Subnetzgruppe aus.
   c. Wählen Sie die Subnetzgruppe aus, die Sie löschen möchten, z. B. example-dual-stack-cluster-subnet-group.
   d. Wählen Sie Löschen, und wählen Sie dann im Bestätigungsfenster Löschen.
2. Notieren Sie sich die VPC-ID.
   a. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc.
   b. Wählen Sie VPC Dashboard und dann Your VPCs aus.
   c. Identifizieren Sie in der Liste die VPC, die Sie erstellt haben, z. B. vpc-example-dual-stack.
   d. Notieren Sie die VPC ID (VPC-ID) der VPC, die Sie erstellt haben. Sie benötigen die VPC-ID in späteren Schritten.
3. Löschen der Sicherheitsgruppen:
   a. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc.
   b. Wählen Sie VPC Dashboard und dann Sicherheitsgruppen.
   c. Wählen Sie die Sicherheitsgruppe für den Amazon DocumentDB-Cluster aus, z. B. example-dual-stack-securitygroup.

d. Wählen Sie für Aktionen die Option Sicherheitsgruppen löschen und dann im Bestätigungsdialogfeld Löschen aus.

e. Zurück auf der Seite „Sicherheitsgruppen“ wählen Sie die Sicherheitsgruppe für die Amazon EC2-Instance aus, z. B. example-securitygroup.

f. Wählen Sie für Aktionen die Option Sicherheitsgruppen löschen und dann im Bestätigungsdialogfeld Löschen aus.

4. Löschen des NAT-Gateways:

a. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc.

b. Wählen Sie VPC Dashboard und dann Sicherheitsgruppen.

c. Wählen Sie das NAT-Gateway der VPC aus, die Sie erstellt haben. Verwenden Sie die VPC-ID, um das richtige NAT-Gateway zu identifizieren.

d. Wählen Sie unter Actions (Aktionen) die Option Delete NAT gateway (NAT-Gateway löschen) aus.

e. Geben Sie im Bestätigungsdialogfeld den Text ein**`delete`**, und wählen Sie dann Löschen aus.

5. Löschen der VPC:

a. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc.

b. Wählen Sie VPC Dashboard und dann Your VPCs aus.

c. Wählen Sie die VPC aus, die Sie löschen möchten, z. B. vpc-example-dual-stack.

d. Wählen Sie für Aktionen die Option VPC löschen aus.

Auf der Bestätigungsseite werden weitere Ressourcen angezeigt, die der VPC zugeordnet sind, die ebenfalls gelöscht werden, einschließlich der damit verknüpften Subnetze.

e. Geben Sie im Bestätigungsdialogfeld den Text ein**`delete`**, und wählen Sie dann Löschen aus.

6. Geben Sie die elastische IP-Adresse frei:

a. Öffnen Sie die EC2-Konsole unter https://console.aws.amazon.com/ec2.

b. Klicken Sie aufEC2-DashboardKlicken Sie auf und danach aufElastische IP-Adressenaus.

c. Wählen Sie die Elastic-IP-Adresse aus, die Sie freigeben möchten.

d. Wählen Sie unter Actions (Aktionen) die Option Release Elastic IP addresses (Elastic-IP-Adressenfreigeben) aus.

e. Wählen Sie im Bestätigungsdialogfeld die Option Release.

Sichern und Wiederherstellen in Amazon DocumentDB

Amazon DocumentDB (mit MongoDB-Kompatibilität) sichert Ihre Daten kontinuierlich für 1—35 Tage auf Amazon Simple Storage Service (Amazon S3), sodass Sie die Daten innerhalb des Aufbewahrungszeitraums für Backups schnell wiederherstellen können. Amazon DocumentDB erstellt im Rahmen dieses kontinuierlichen Sicherungsprozesses auch automatische Schnappschüsse Ihrer Daten.

> **Note**
>
> Dabei handelt es sich um vom Service verwaltete Amazon S3-Buckets, und Sie haben keinen Zugriff auf die Sicherungsdateien. Informationen zur Steuerung Ihrer eigenen Backups finden Sie unter. Daten speichern, wiederherstellen, importieren und exportieren

Sie können Backup-Daten auch über die Aufbewahrungsfrist hinaus aufbewahren, indem Sie einen manuellen Snapshot der Daten Ihres Clusters erstellen. Der Backup-Prozess hat keinen Einfluss auf die Leistung Ihres Clusters.

In diesem Abschnitt werden die Anwendungsfälle der Backup-Funktionen in Amazon DocumentDB beschrieben und es wird Ihnen gezeigt, wie Sie Backups für Ihre Amazon DocumentDB-Cluster verwalten.

Themen

- Sichern und Wiederherstellen: Konzepte
- Grundlegendes zur Sicherungsspeicher-Nutzung in
- Daten speichern, wiederherstellen, importieren und exportieren
- Überlegungen zum Cluster-Snapshot
- Vergleich von automatischen und manuellen Snapshots
- Einen manuellen Cluster-Snapshot erstellen
- Kopieren von Amazon DocumentDB-Cluster-Snapshots
- Amazon DocumentDB-Cluster-Snapshots teilen
- Wiederherstellung aus einem Cluster-Snapshot
- Wiederherstellung auf einen bestimmten Zeitpunkt

- Löschen eines Cluster-Snapshots

Sichern und Wiederherstellen: Konzepte

Substantiv	Description	APIs (Verben)

Aufbewahrungszeitraum für Backups	Ein Zeitraum zwischen 1 und 35 Tagen, für den Sie eine Point-in-Time Wiederherstellung durchführen können.	`create-db-cluster`
		`modify-db-cluster`
		`restore-db-cluster-to-point-in-time`

Amazon DocumentDB-Speichervolumen	Hochverfügbares und sehr robustes Speichervolume, das Daten auf sechs Arten in drei Availability Zones repliziert. Ein Amazon DocumentDB-Cluster	`create-db-cluster` `delete-db-cluster`

Substantiv	Description	APIs (Verben)
	ist unabhängig von der Anzahl der Instances im Cluster äußerst robust.	

Backup window	Zeitspanne an dem Tag, an dem automatische Snapshots gemacht werden.	`create-db-cluster`
		`describe-db-cluster`
		`modify-db-cluster`

Substantiv	Description	APIs (Verben)

Automatischer Snapshot	Tägliche Snapshots, bei denen es sich um vollständige Backups des Clusters handelt und die automatisch durch den kontinuierlichen Backup-Prozess in Amazon DocumentDB erstellt werden.	`restore-db-cluster-from-snapshot`
		
		`describe-db-cluster-snapshot-attributes`
		
		`describe-db-cluster-snapshots`

Substantiv	Description	APIs (Verben)

Manueller Snapshot	Snapshots, die Sie manuell erstellen, um Vollsicherungen eines Clusters über den Sicherungszeitraum hinaus aufzubewahren.	`create-db-cluster-snapshot`
		`copy-db-cluster-snapshot`
		`delete-db-cluster-snapshot`
		`describe-db-cluster-snapshot-attributes`
		`describe-db-cluster-snapshots`
		`modify-db-cluster-snapshot-attribute`

Grundlegendes zur Sicherungsspeicher-Nutzung in

Der Amazon DocumentDB-Backup-Speicher besteht aus kontinuierlichen Backups innerhalb des Aufbewahrungszeitraums für Backups und manuellen Snapshots außerhalb des Aufbewahrungszeitraums. Wenn Sie Ihren Sicherungsspeicher kontrollieren möchten, können Sie das Aufbewahrungsintervall für Sicherungen verringern, alte (nicht mehr benötigte) manuelle Snapshots entfernen oder beides. Allgemeine Informationen zu Amazon DocumentDB-Backups finden Sie unter. Sichern und Wiederherstellen in Amazon DocumentDB Preisinformationen zum Amazon DocumentDB-Backup-Speicher finden Sie unter Amazon DocumentDB-Preise.

Wenn Sie Ihre Kosten kontrollieren möchten, können Sie die Menge des Speichers überwachen, der nach Ablauf des Aufbewahrungszeitraums noch von kontinuierlichen Sicherungen und manuellen Snapshots belegt wird. Anschließend können Sie den Aufbewahrungszeitraum für Backups verringern und manuelle Snapshots entfernen, wenn sie nicht mehr benötigt werden.

Sie können die CloudWatch Amazon-Metriken `TotalBackupStorageBilled` verwenden und `BackupRetentionPeriodStorageUsed` den von Ihren Amazon DocumentDB-Backups verwendeten Speicherplatz wie folgt überprüfen und überwachen: `SnapshotStorageUsed`

- `BackupRetentionPeriodStorageUsed` stellt den Umfang des Backup-Speichers zum Speichern von kontinuierlichen Backups zur aktuellen Zeit dar. Dieser Metrikwert hängt von der Größe des Cluster-Volumens und der Menge der von Ihnen während des Aufbewahrungszeitraums vorgenommenen Änderungen ab. Für Fakturierungszwecke übersteigt die Metrik die kumulative Cluster-Volume-Größe während des Aufbewahrungszeitraums nicht. Beispiel: Wenn Ihr Cluster 100 GiB groß ist und Ihr Aufbewahrungszeitraum zwei Tage beträgt, hat `BackRetentionPeriodStorageUsed` einen Maximalwert von 200 GiB (100 GiB+100 GiB).

- `SnapshotStorageUsed` stellt den Umfang des Backup-Speichers zum Speichern von manuellen Snapshots außerhalb des Aufbewahrungszeitraums für Backups dar. Innerhalb des Aufbewahrungszeitraums manuell aufgenommene Snapshots zählen nicht zu Ihrem Backup-Speicher. Ebenso werden automatische Snapshots nicht zu Ihrem Backup-Speicher gezählt. Die Größe einzelner Snapshots entspricht der Größe des Cluster-Volumes zum Zeitpunkt der Snapshot-Erstellung. Der `SnapshotStorageUsed`-Wert hängt von der Anzahl der Snapshots ab, die Sie aufbewahren, und von der Größe der einzelnen Snapshots. Beispiel: Angenommen, Sie haben einen Snapshot außerhalb des Aufbewahrungszeitraums und die Cluster-Volume-Größe betrug 100 GiB, als der Snapshot erstellt wurde. Dann beträgt die `SnapshotStorageUsed`-Menge 100 GiB.

- `TotalBackupStorageBilled` stellt die Summe von `BackupRetentionPeriodStorageUsed` und `SnapshotStorageUsed` abzüglich des freien Backup-Speichers gleich der Größe des Cluster-Volumes für einen Tag dar. Wenn Ihre Clustergröße beispielsweise 100 GiB beträgt, Sie einen Aufbewahrungstag haben und Sie einen Snapshot außerhalb des Aufbewahrungszeitraums haben, sind `TotalBackupStorageBilled` es 100 GiB (100 GiB + 100 GiB — 100 GiB).

- Diese Metriken werden unabhängig für jeden Amazon DocumentDB-Cluster berechnet.

Über die Konsole können Sie Ihre Amazon DocumentDB-Cluster überwachen und anhand von CloudWatch Metriken Berichte erstellen. CloudWatch Weitere Informationen zur Verwendung von CloudWatch Metriken finden Sie unterÜberwachung und Protokollierung in Amazon DocumentDB.

Daten speichern, wiederherstellen, importieren und exportieren

Sie können die `mongoimport` Dienstprogrammemongodump,`mongorestore`, und verwendenmongoexport, um Daten in und aus Ihrem Amazon DocumentDB-Cluster zu verschieben. In diesem Abschnitt wird der Zweck der einzelnen Tools und Konfigurationen beschrieben, damit Sie die Leistung verbessern können.

> **Versionsanforderung für die MongoDB-Datenbanktools**
>
> Verwenden Sie die MongoDB-Datenbanktools bis einschließlich Version 100.11.0 für Amazon DocumentDB. Informationen zum Herunterladen finden Sie in den Versionen der MongoDB Database Tools auf der MongoDB-Website.

Themen

- Mongodump
- mongorestore
- Mongoexport
- Mongoimport
- Tutorial

Mongodump

Das Dienstprogramm `mongodump` erstellt eine binäre (BSON)-Sicherung einer MongoDB-Datenbank. Das `mongodump` Tool ist aufgrund der Größeneffizienz, die durch das Speichern der Daten in einem Binärformat erreicht wird, die bevorzugte Methode zum Dumping von Daten aus Ihrer MongoDB-Quellbereitstellung, wenn Sie sie in Ihrem Amazon DocumentDB-Cluster wiederherstellen möchten.

Abhängig von den Ressourcen, die auf der Instance oder dem Computer verfügbar sind, den Sie für die Ausführung des Befehls verwenden, können Sie Ihren Vorgang beschleunigen, `mongodump` indem Sie die Anzahl der parallelen Sammlungen, die standardmäßig gespeichert werden, mithilfe der Option 1 erhöhen. `--numParallelCollections` Eine gute Faustregel lautet, mit einem Worker pro vCPU auf der primären Instance Ihres Amazon DocumentDB-Clusters zu beginnen.

Beispielverwendung

Im Folgenden finden Sie ein Beispiel für die Verwendung des mongodump Dienstprogramms im Amazon DocumentDB-Cluster,. sample-cluster

```
mongodump --ssl \
    --host="sample-cluster.node.us-east-1.docdb.amazonaws.com:27017" \
    --collection=sample-collection \
    --db=sample-database \
    --out=sample-output-file \
    --numParallelCollections 4  \
    --username=sample-user \
    --password=abc0123 \
    --sslCAFile global-bundle.pem
```

mongorestore

Mit dem Dienstprogramm mongorestore können Sie eine Binärsicherung (BSON) einer Datenbank wiederherstellen, die mit dem Dienstprogramm mongodump erstellt wurde. Sie können die Wiederherstellungsleistung verbessern, indem Sie die Anzahl der Worker für jede Sammlung während der Wiederherstellung mit der Option --numInsertionWorkersPerCollection erhöhen. (Der Standardwert ist 1.) Eine gute Faustregel ist, mit einem Worker pro vCPU auf der primären Instance Ihres Amazon DocumentDB-Clusters zu beginnen.

Beispielverwendung

Im Folgenden finden Sie ein Beispiel für die Verwendung des mongorestore Dienstprogramms im Amazon DocumentDB-Cluster,. sample-cluster

```
mongorestore --ssl \
    --host="sample-cluster.node.us-east-1.docdb.amazonaws.com:27017" \
    --username=sample-user \
    --password=abc0123 \
    --sslCAFile global-bundle.pem <fileToBeRestored>
```

Mongoexport

Das mongoexport Tool exportiert Daten in Amazon DocumentDB in die Dateiformate JSON, CSV oder TSV. Das mongoexport-Tool ist die bevorzugte Methode für den Export von Daten, die für Menschen oder Maschinen lesbar sein müssen.

> **Note**
>
> `mongoexport` unterstützt parallele Exporte nicht direkt. Es ist jedoch möglich, die Leistung zu steigern, indem mehrere `mongoexport`-Aufgaben gleichzeitig für verschiedene Sammlungen ausgeführt werden.

Beispielverwendung

Im Folgenden finden Sie ein Beispiel für die Verwendung des `mongoexport` Tools im Amazon DocumentDB-Cluster. `sample-cluster`

```
mongoexport --ssl \
    --host="sample-cluster.node.us-east-1.docdb.amazonaws.com:27017" \
    --collection=sample-collection \
    --db=sample-database \
    --out=sample-output-file \
    --username=sample-user \
    --password=abc0123 \
    --sslCAFile global-bundle.pem
```

`Mongoimport`

Das `mongoimport` Tool importiert den Inhalt von JSON-, CSV- oder TSV-Dateien in einen Amazon DocumentDB-Cluster. Mithilfe des Parameters `--numInsertionWorkers` können Sie den Import parallelisieren und beschleunigen. (Der Standardwert ist 1.)

Beispielverwendung

Im Folgenden finden Sie ein Beispiel für die Verwendung des `mongoimport` Tools im Amazon DocumentDB-Cluster. `sample-cluster`

```
mongoimport --ssl \
    --host="sample-cluster.node.us-east-1.docdb.amazonaws.com:27017" \
    --collection=sample-collection \
    --db=sample-database \
    --file=<yourFile> \
    --numInsertionWorkers 4 \
    --username=sample-user \
```

```
    --password=abc0123 \
    --sslCAFile global-bundle.pem
```

Tutorial

Das folgende Tutorial beschreibt, wie Sie die `mongoimport` Dienstprogramme`mongodump`,, und verwenden `mongorestore`mongoexport, um Daten in einen Amazon DocumentDB-Cluster hinein und aus diesem zu verschieben.

1. Voraussetzungen — Bevor Sie beginnen, stellen Sie sicher, dass Ihr Amazon DocumentDB-Cluster bereitgestellt ist und dass Sie Zugriff auf eine Amazon EC2-Instance in derselben VPC wie Ihr Cluster haben. Weitere Informationen finden Sie unter Stellen Sie mithilfe von Amazon EC2 eine Verbindung her.

   Um die mongo-Dienstprogrammtools verwenden zu können, muss das mongodb-org-tools-Paket wie folgt in Ihrer EC2-Instance installiert sein.

   ```
   sudo yum install mongodb-org-tools-4.0.18
   ```

   Da Amazon DocumentDB standardmäßig die Transport Layer Security (TLS) -Verschlüsselung verwendet, müssen Sie auch die Amazon RDS-Zertifizierungsstelle (CA) herunterladen, um die Mongo-Shell für die Verbindung wie folgt zu verwenden.

   ```
   wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
   ```

2. Beispieldaten herunterladen — Für dieses Tutorial laden Sie einige Beispieldaten herunter, die Informationen über Restaurants enthalten.

   ```
   wget https://raw.githubusercontent.com/ozlerhakan/mongodb-json-files/master/
   datasets/restaurant.json
   ```

3. Importieren Sie die Beispieldaten in Amazon DocumentDB — Da die Daten in einem logischen JSON-Format vorliegen, verwenden Sie das `mongoimport` Hilfsprogramm, um die Daten in Ihren Amazon DocumentDB-Cluster zu importieren.

   ```
   mongoimport --ssl \
       --host="tutorialCluster.amazonaws.com:27017" \
       --collection=restaurants \
       --db=business \
       --file=restaurant.json \
   ```

```
    --numInsertionWorkers 4 \
    --username=<yourUsername> \
    --password=<yourPassword> \
    --sslCAFile global-bundle.pem
```

4. Daten speichern mit **`mongodump`** — Da Sie nun Daten in Ihrem Amazon DocumentDB-Cluster haben, können Sie mithilfe des Hilfsprogramms einen binären Speicherauszug dieser Daten erstellen. `mongodump`

```
mongodump --ssl \
    --host="tutorialCluster.us-east-1.docdb.amazonaws.com:27017"\
    --collection=restaurants \
    --db=business \
    --out=restaurantDump.bson \
    --numParallelCollections 4 \
    --username=<yourUsername> \
    --password=<yourPassword> \
    --sslCAFile global-bundle.pem
```

5. **`restaurants`**Sammlung löschen — Bevor Sie die `restaurants` Sammlung in der `business` Datenbank wiederherstellen, müssen Sie zunächst die Sammlung löschen, die bereits in dieser Datenbank vorhanden ist, wie folgt.

```
use business
```

```
db.restaurants.drop()
```

6. Daten wiederherstellen mit **`mongorestore`** — Mit dem binären Dump der Daten aus Schritt 3 können Sie jetzt das `mongorestore` Tool verwenden, um Ihre Daten in Ihrem Amazon DocumentDB-Cluster wiederherzustellen.

```
mongorestore --ssl \
    --host="tutorialCluster.us-east-1.docdb.amazonaws.com:27017" \
    --numParallelCollections 4 \
    --username=<yourUsername> \
    --password=<yourPassword> \
    --sslCAFile global-bundle.pem restaurantDump.bson
```

7. Exportieren Sie die Daten mit **`mongoexport`** — Um das Tutorial abzuschließen, exportieren Sie die Daten aus Ihrem Cluster im Format einer JSON-Datei, die sich nicht von der Datei unterscheidet, die Sie in Schritt 1 importiert haben.

```
mongoexport --ssl \
    --host="tutorialCluster.node.us-east-1.docdb.amazonaws.com:27017" \
    --collection=restaurants \
    --db=business \
    --out=restaurant2.json \
    --username=<yourUsername> \
    --password=<yourPassword> \
    --sslCAFile global-bundle.pem
```

8. Validierung — Mit den folgenden Befehlen können Sie überprüfen, ob die Ausgabe von Schritt 5 dasselbe Ergebnis wie Schritt 1 liefert.

```
wc -l restaurant.json
```

Ausgabe dieses Befehls:

```
2548 restaurant.json
```

```
wc -l restaurant2.json
```

Ausgabe dieses Befehls:

```
2548 restaurant2.json
```

Überlegungen zum Cluster-Snapshot

Amazon DocumentDB erstellt täglich automatische Snapshots Ihres Clusters während des Backup-Fensters Ihres Clusters. Amazon DocumentDB speichert die automatischen Snapshots Ihres Clusters gemäß dem von Ihnen angegebenen Aufbewahrungszeitraum für Backups. Bei Bedarf können Sie Ihren Cluster zu einem beliebigen Zeitpunkt während der Aufbewahrungsdauer des Backups wiederherstellen. Automatische Snapshots treten nicht auf, während ein Kopiervorgang in derselben Region für denselben Cluster ausgeführt wird.

Themen

- Backup-Speicher
- Backup window

- Aufbewahrungszeitraum für Backups
- Cluster-Snapshot-Verschlüsselung kopieren

Zusätzlich zu den automatischen Cluster-Snapshots können Sie auch manuell einen Cluster-Snapshot erstellen. Sie können sowohl automatische als auch manuelle Snapshots kopieren. Weitere Informationen erhalten Sie unter Einen manuellen Cluster-Snapshot erstellen und Kopieren von Amazon DocumentDB-Cluster-Snapshots.

> **Note**
>
> Ihr Cluster muss sich im Status verfügbar befinden, damit ein automatischer Snapshot erstellt werden kann.
> Sie können einen automatisierten Cluster-Snapshot von Amazon DocumentDB nicht teilen. Als Problemumgehung können Sie einen manuellen Snapshot erstellen, indem Sie den automatisierten Snapshot kopieren und dann diese Kopie freigeben. Weitere Informationen zum Kopieren eines Snapshots finden Sie unter Kopieren von Amazon DocumentDB-Cluster-Snapshots. Weitere Informationen zum Wiederherstellen eines Clusters aus einem Snapshot finden Sie unter Wiederherstellung aus einem Cluster-Snapshot.

Backup-Speicher

Ihr Amazon DocumentDB-Backup-Speicher für jeden AWS-Region besteht aus dem Backup-Speicher, der für Ihren Backup-Aufbewahrungszeitraum benötigt wird. Dazu gehören automatische und manuelle Cluster-Snapshots in dieser Region. Der standardmäßige Aufbewahrungszeitraum für Backups beträgt 1 Tag. Weitere Informationen zu den Preisen für Backup-Speicher finden Sie unter Amazon DocumentDB-Preise.

Wenn Sie einen Cluster löschen, werden alle seine automatischen Snapshots gelöscht und können nicht wiederhergestellt werden. Manuelle Snapshots werden jedoch nicht gelöscht, wenn Sie einen Cluster löschen. Wenn Sie sich dafür entscheiden, dass Amazon DocumentDB einen endgültigen Snapshot (manueller Snapshot) erstellt, bevor Ihr Cluster gelöscht wird, können Sie den endgültigen Snapshot verwenden, um Ihren Cluster wiederherzustellen.

Weitere Informationen zu Snapshots und Speicher finden Sie unter Grundlegendes zur Sicherungsspeicher-Nutzung in.

Backup window

Automatische Snapshots erfolgen täglich während des bevorzugten Backup-Fensters. Wenn der Snapshot mehr Zeit benötigt, als dem Backup-Fenster zugewiesen wurde, wird der Backup-Prozess bis zum Ende fortgesetzt, auch wenn das Backup-Fenster beendet ist. Das Backup-Fenster kann sich nicht mit dem wöchentlichen Wartungsfenster für den Cluster überschneiden.

Wenn Sie bei der Erstellung des Clusters kein bevorzugtes Backup-Fenster angeben, weist Amazon DocumentDB standardmäßig ein 30-minütiges Backup-Fenster zu. Dieses Fenster wird nach dem Zufallsprinzip aus einem 8-stündigen Zeitblock ausgewählt, der mit der Region Ihres Clusters verknüpft ist. Sie können Ihr bevorzugtes Backup-Fenster ändern, indem Sie den Cluster ändern. Weitere Informationen finden Sie unter Ändern eines Amazon DocumentDB-Clusters.

Name der Region	Region	UTC-Zeitblock
USA Ost (Ohio)	us-east-2	03:00-11:00
USA Ost (Nord-Virginia)	us-east-1	03:00-11:00
USA West (Oregon)	us-west-2	06:00-14:00
Afrika (Kapstadt)	af-south-1	03:00 — 11:00
Asien-Pazifik (Hongkong)	ap-east-1	06:00-14:00
Asien-Pazifik (Hyderabad)	ap-south-2	06:30 — 14:30
Asien-Pazifik (Malaysia)	ap-southeast-5	13:00-21:00
Asien-Pazifik (Mumbai)	ap-south-1	06:00-14:00
Asia Pacific (Osaka)	ap-northeast-3	12:00-20:00
Asien-Pazifik (Seoul)	ap-northeast-2	13:00-21:00
Asien-Pazifik (Singapur)	ap-southeast-1	14:00-22:00
Asien-Pazifik (Sydney)	ap-southeast-2	12:00-20:00
Asien-Pazifik (Jakarta)	ap-southeast-3	08:00-16:00

Name der Region	Region	UTC-Zeitblock
Asien-Pazifik (Melbourne)	ap-southeast-4	11:00-19:00
Asien-Pazifik (Thailand)	ap-southeast-7	15:00-23:00
Asien-Pazifik (Tokio)	ap-northeast-1	13:00-21:00
Kanada (Zentral)	ca-central-1	03:00-11:00
Kanada West (Calgary)	ca-west-1	18:00-02:00
China (Beijing)	cn-north-1	06:00-14:00
China (Ningxia)	cn-northwest-1	06:00-14:00
Europa (Frankfurt)	eu-central-1	21:00-05:00
Europa (Zürich)	eu-central-2	02:00-10:00
Europa (Irland)	eu-west-1	22:00-06:00
Europa (London)	eu-west-2	22:00-06:00
Europa (Milan)	eu-south-1	02:00-10:00
Europa (Paris)	eu-west-3	23:59-07:29
Europa (Spain)	eu-south-2	02:00 — 10:00
Europa (Stockholm)	eu-north-1	04:00 — 12:00
Mexiko (Zentral)	mx-central-1	03:00-11:00
Naher Osten (VAE)	me-central-1	05:00 — 13:00
Südamerika (São Paulo)	sa-east-1	00:00-08:00
Israel (Tel Aviv)	il-central-1	04:00-12:00
AWS GovCloud (US-East)	us-gov-east-1	17:00-01:00

Name der Region	Region	UTC-Zeitblock
AWS GovCloud (US-West)	us-gov-west-1	06:00-14:00

Aufbewahrungszeitraum für Backups

Die Aufbewahrungsdauer für Backups ist die Anzahl der Tage, an denen ein automatisches Backup aufbewahrt wird, bevor es automatisch gelöscht wird. Amazon DocumentDB unterstützt eine Aufbewahrungsfrist für Backups von 1—35 Tagen.

Sie können die Aufbewahrungsdauer des Backups beim Anlegen eines Clusters festlegen. Wenn Sie die Aufbewahrungsdauer für das Backup nicht explizit festlegen, wird Ihrem Cluster die standardmäßige Aufbewahrungsdauer für das Backup von 1 Tag zugewiesen. Nachdem Sie einen Cluster erstellt haben, können Sie den Aufbewahrungszeitraum für Backups ändern, indem Sie den Cluster entweder mit AWS-Managementkonsole oder ändern. AWS CLI Weitere Informationen finden Sie unter Ändern eines Amazon DocumentDB-Clusters.

Cluster-Snapshot-Verschlüsselung kopieren

Die Cluster- und Snapshot-Verschlüsselung basiert auf einem KMS-Verschlüsselungsschlüssel. Die KMS-Schlüssel-ID ist der Amazon-Ressourcenname (ARN), der KMS-Schlüsselbezeichner oder der KMS-Schlüsselalias für den KMS-Verschlüsselungsschlüssel.

Es gelten die folgenden Richtlinien und Einschränkungen:

- Bei der Erstellung eines Snapshots wird die Verschlüsselung aus dem Cluster abgeleitet. Wenn der Cluster verschlüsselt ist, wird der Snapshot dieses Clusters mit demselben KMS-Schlüssel verschlüsselt. Wenn der Cluster nicht verschlüsselt ist, ist der Snapshot nicht verschlüsselt.
- Wenn Sie einen verschlüsselten Cluster-Snapshot von Ihrem Amazon Web Services-Konto kopieren, können Sie einen Wert für angeben, `KmsKeyId` um die Kopie mit einem neuen KMS-Verschlüsselungsschlüssel zu verschlüsseln. Wenn Sie keinen Wert für angeben`KmsKeyId`, wird die Kopie des Cluster-Snapshots mit demselben KMS-Schlüssel wie der Quell-Cluster-Snapshot verschlüsselt.
- Wenn Sie einen verschlüsselten Cluster-Snapshot kopieren, der von einem anderen Amazon Web Services-Konto gemeinsam genutzt wird, müssen Sie einen Wert für angeben`KmsKeyId`.
- Um einen verschlüsselten Cluster-Snapshot in eine andere Amazon Web Services-Region `KmsKeyId` zu kopieren, geben Sie die KMS-Schlüssel-ID ein, die Sie zum Verschlüsseln der Kopie

des Cluster-Snapshots in der Zielregion verwenden möchten. KMS-Verschlüsselungsschlüssel sind spezifisch für die Amazon Web Services-Region, in der sie erstellt wurden, und Sie können keine Verschlüsselungsschlüssel aus einer Amazon Web Services-Region in einer anderen Amazon Web Services-Region verwenden.

- Wenn Sie einen unverschlüsselten Cluster-Snapshot kopieren und einen Wert für den `KmsKeyId` Parameter angeben, wird ein Fehler zurückgegeben.

Vergleich von automatischen und manuellen Snapshots

Im Folgenden sind die wichtigsten Funktionen der automatischen und manuellen Snapshots von Amazon DocumentDB (mit MongoDB-Kompatibilität) aufgeführt.

Die automatischen Snapshots von Amazon DocumentDB verfügen über die folgenden Hauptfunktionen:

- Automatische Snapshot-Benennung — Automatische Snapshot-Namen folgen dem Muster und `yyyy-mm-dd-hh-mm` stellen das Datum und die Uhrzeit dar`rds:<cluster-name>-yyyy-mm-dd-hh-mm`, zu der der Snapshot erstellt wurde.
- Automatisch nach einem Zeitplan erstellt — Wenn Sie einen Cluster erstellen oder ändern, können Sie den Aufbewahrungszeitraum für Backups auf einen Ganzzahlwert zwischen 1 und 35 Tagen festlegen. Standardmäßig haben neue Cluster eine Aufbewahrungsdauer von 1 Tag. Die Aufbewahrungsdauer für Backups definiert die Anzahl der Tage, die automatische Snapshots aufbewahrt werden, bevor sie automatisch gelöscht werden. Sie können automatische Backups auf Amazon DocumentDB-Clustern nicht deaktivieren.

  Zusätzlich zur Einstellung der Aufbewahrungsdauer für das Backup legen Sie auch das Backup-Fenster fest (die Tageszeit, zu der automatische Snapshots erstellt werden).
- Automatische Snapshots löschen — Automatische Snapshots werden gelöscht, wenn Sie den Cluster des automatischen Snapshots löschen. Sie können einen automatischen Snapshot nicht manuell löschen.
- Inkrementell — Während des Aufbewahrungszeitraums für Backups werden Datenbankaktualisierungen aufgezeichnet, sodass die Änderungen schrittweise aufgezeichnet werden.
- Wiederherstellung aus einem automatischen Snapshot — Sie können aus einem automatischen Snapshot wiederherstellen, indem Sie die AWS-Managementkonsole oder verwenden. AWS CLI

Wenn Sie die Wiederherstellung aus einem Snapshot mithilfe von AWS CLI durchführen, müssen Sie Instanzen separat hinzufügen, sobald der Cluster verfügbar ist.

- Teilen — Sie können einen automatisierten Cluster-Snapshot von Amazon DocumentDB nicht teilen. Als Problemumgehung können Sie einen manuellen Snapshot erstellen, indem Sie den automatisierten Snapshot kopieren und dann diese Kopie freigeben. Weitere Informationen zum Kopieren eines Snapshots finden Sie unter Kopieren von Amazon DocumentDB-Cluster-Snapshots. Weitere Informationen zum Wiederherstellen eines Clusters aus einem Snapshot finden Sie unter Wiederherstellung aus einem Cluster-Snapshot.
- Sie können die Daten von jedem Zeitpunkt innerhalb des Aufbewahrungszeitraums für Backups aus wiederherstellen — Da Datenbankaktualisierungen inkrementell aufgezeichnet werden, können Sie Ihren Cluster zu jedem beliebigen Zeitpunkt innerhalb des Aufbewahrungszeitraums für Backups wiederherstellen.

  Wenn Sie die Wiederherstellung aus einem automatischen Snapshot oder aus einer Point-in-Time-Wiederherstellung mithilfe von durchführen AWS CLI, müssen Sie Instanzen separat hinzufügen, sobald der Cluster verfügbar ist.

Manuelle Amazon DocumentDB-Snapshots haben die folgenden Hauptfunktionen:

- Bei Bedarf erstellt — Manuelle Amazon DocumentDB-Snapshots werden bei Bedarf mithilfe der Amazon DocumentDB Management Console oder erstellt. AWS CLI
- Löschen eines manuellen Snapshots — Ein manueller Snapshot wird nur gelöscht, wenn Sie ihn explizit entweder mit der Amazon DocumentDB-Konsole löschen oder. AWS CLI Ein manueller Snapshot wird nicht gelöscht, wenn Sie seinen Cluster löschen.
- Vollständige Backups — Wenn ein manueller Snapshot erstellt wird, wird eine vollständige Sicherung der Daten Ihres Clusters erstellt und gespeichert.
- Manuelle Snapshot-Benennung — Sie geben den manuellen Snapshot-Namen an. Amazon DocumentDB fügt dem Namen keinen `datetime` Stempel hinzu. Sie müssen diese Informationen also hinzufügen, wenn Sie möchten, dass sie im Namen enthalten sind.
- Wiederherstellung aus einem manuellen Snapshot — Sie können die Wiederherstellung aus einem manuellen Snapshot mithilfe der Konsole oder der durchführen. AWS CLI Wenn Sie die Wiederherstellung aus einem Snapshot mithilfe von AWS CLI durchführen, müssen Sie Instanzen separat hinzufügen, sobald der Cluster verfügbar ist.
- Servicekontingente — Sie sind auf maximal 100 manuelle Snapshots pro Person AWS-Region beschränkt.

- Teilen — Sie können manuelle Cluster-Snapshots teilen, die von autorisierten Personen kopiert werden können. AWS-Konten Sie können verschlüsselte oder unverschlüsselte manuelle Snapshots freigeben. Weitere Informationen zum Kopieren eines Snapshots finden Sie unter Kopieren von Amazon DocumentDB-Cluster-Snapshots.
- Sie stellen den Zeitpunkt wieder her, an dem der manuelle Snapshot erstellt wurde — Wenn Sie die Wiederherstellung anhand eines manuellen Snapshots durchführen, stellen Sie den Zeitpunkt wieder her, an dem der manuelle Snapshot erstellt wurde.

  Wenn Sie die Wiederherstellung aus einem Snapshot mithilfe von AWS CLI durchführen, müssen Sie Instanzen separat hinzufügen, sobald der Cluster verfügbar ist.

Einen manuellen Cluster-Snapshot erstellen

Sie können einen manuellen Snapshot entweder mit AWS-Managementkonsole oder AWS CLI erstellen. Die Zeit, die für die Erstellung eines Snapshots benötigt wird, hängt von der Größe Ihrer Datenbanken ab. Wenn Sie einen Snapshot erstellen, müssen Sie Folgendes tun:

1. Identifizieren Sie den zu sichernden Cluster.
2. Geben Sie Ihrem Snapshot einen Namen. Dies ermöglicht es Ihnen, ihn später wiederherzustellen.

Using the AWS-Managementkonsole

Um einen manuellen Snapshot mit dem zu erstellen AWS-Managementkonsole, können Sie eine der folgenden Methoden verwenden.

1. Methode 1:

   1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
   2. Wählen Sie im Navigationsbereich die Option Snapshots.

      > **Tip**
      >
      > Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Wählen Sie auf der Seite Snapshots die Option Create (Erstellen) aus.
4. Auf der Seite Create cluster snapshot (Cluster-Snapshot erstellen):
   a. Cluster-ID — Wählen Sie aus der Dropdownliste der Cluster den Cluster aus, von dem Sie einen Snapshot erstellen möchten.
   b. Snapshot-ID — Geben Sie einen Namen für Ihren Snapshot ein.

      Einschränkungen bei der Snapshot-Benennung:

      - Die Länge besteht aus [1—255] Buchstaben, Zahlen oder Bindestrichen.
      - Muss mit einem Buchstaben beginnen.
      - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.
      - Muss für alle Cluster (in Amazon RDS, Amazon Neptune und Amazon DocumentDB) pro Konto und Region eindeutig sein. AWS
   c. Wählen Sie Erstellen aus.

2. Methode 2:

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Klicken Sie im Navigationsbereich auf Cluster.

> **Tip**
>
> Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Wählen Sie auf der Seite Cluster die Schaltfläche links neben dem Cluster aus, den Sie als Snapshot aufnehmen möchten.
4. Wählen Sie im Menü Actions (Aktionen) die Option Take snapshot (Snapshot erstellen) aus.
5. Auf der Seite Create cluster snapshot (Cluster-Snapshot erstellen):
   a. Snapshot-ID — Geben Sie einen Namen für Ihren Snapshot ein.

      Einschränkungen bei der Snapshot-Benennung:

      - Die Länge besteht aus [1—63] Buchstaben, Zahlen oder Bindestrichen.

- Muss mit einem Buchstaben beginnen.
- Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.
- Muss für alle Cluster (in Amazon RDS, Amazon Neptune und Amazon DocumentDB) pro Konto und Region eindeutig sein. AWS

b. Wählen Sie Erstellen aus.

Using the AWS CLI

Verwenden Sie die `create-db-cluster-snapshot` Operation mit den folgenden Parametern AWS CLI, um einen Cluster-Snapshot mit den folgenden Parametern zu erstellen.

Parameters

- **`--db-cluster-identifier`** – Erforderlich. Der Name des Clusters, von dem Sie einen Snapshot machen. Dieser Cluster muss vorhanden sein und verfügbar sein.
- **`--db-cluster-snapshot-identifier`** – Erforderlich. Der Name des manuellen Snapshots, den Sie erstellen.

Im folgenden Beispiel wird ein Snapshot mit dem Namen `sample-cluster-snapshot` für einen Clusters namens `sample-cluster` erstellt.

Für Linux, macOS oder Unix:

```
aws docdb create-db-cluster-snapshot \
    --db-cluster-identifier sample-cluster \
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

Für Windows:

```
aws docdb create-db-cluster-snapshot ^
    --db-cluster-identifier sample-cluster ^
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
    "DBClusterSnapshot": {
```

```
        "AvailabilityZones": [
            "us-east-1a",
            "us-east-1b",
            "us-east-1c"
        ],
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot",
        "DBClusterIdentifier": "sample-cluster",
        "SnapshotCreateTime": "2020-04-24T04:59:08.475Z",
        "Engine": "docdb",
        "Status": "creating",
        "Port": 0,
        "VpcId": "vpc-abc0123",
        "ClusterCreateTime": "2020-01-10T22:13:38.261Z",
        "MasterUsername": "master-user",
        "EngineVersion": "4.0.0",
        "SnapshotType": "manual",
        "PercentProgress": 0,
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/sample-key",
        "DBClusterSnapshotArn": "arn:aws:rds:us-east-1:<accountID>:cluster-
snapshot:sample-cluster-snapshot"
    }
}
```

Kopieren von Amazon DocumentDB-Cluster-Snapshots

In Amazon DocumentDB können Sie Schnappschüsse innerhalb derselben AWS-Region oder in eine andere Datei kopieren. AWS-Region Sie können auch geteilte Schnappschüsse auf Ihr Konto kopieren, das sich im selben AWS-Region oder in einem anderen Konto befindet. AWS-Region Weitere Informationen zum Freigeben von Snapshots finden Sie unter Amazon DocumentDB-Cluster-Snapshots teilen.

> **Note**
>
> Amazon DocumentDB stellt Ihnen die Rechnung auf der Grundlage der Menge der Backup- und Snapshot-Daten, die Sie aufbewahren, und dem Zeitraum, in dem Sie sie aufbewahren. Weitere Informationen zum Speicherplatz für Amazon DocumentDB-Backups und -Snapshots finden Sie unter. Grundlegendes zur Sicherungsspeicher-Nutzung in Preisinformationen zum Amazon DocumentDB-Speicher finden Sie unter Amazon DocumentDB-Preise.

Themen

- Kopieren freigegebener Snapshots
- Schnappschüsse kopieren AWS-Regionen
- Einschränkungen
- Umgang mit Verschlüsselungen
- Überlegungen zu Parametergruppen
- Einen Cluster-Snapshot kopieren

Kopieren freigegebener Snapshots

Sie können Schnappschüsse kopieren, die Ihnen von anderen Konten zur Verfügung gestellt wurden. AWS Wenn Sie einen verschlüsselten Snapshot kopieren, der von einem anderen AWS Konto aus geteilt wurde, müssen Sie Zugriff auf den AWS KMS Verschlüsselungsschlüssel haben, der zum Verschlüsseln des Snapshots verwendet wurde. Weitere Informationen finden Sie unter Umgang mit Verschlüsselungen.

Schnappschüsse kopieren AWS-Regionen

Wenn Sie einen Snapshot in einen AWS-Region anderen als den Quell-Snapshot kopieren AWS-Region, ist jede Kopie ein vollständiger Snapshot. Eine vollständige Snapshot-Kopie enthält alle Daten und Metadaten, die für die Wiederherstellung des Amazon DocumentDB-Clusters erforderlich sind.

Einschränkungen

Im folgenden werden einige Einschränkungen beim Kopieren von Snapshots aufgeführt:

- Wenn Sie einen Quell-Snapshot löschen, bevor der Ziel-Snapshot verfügbar ist, kann das Kopieren des Snapshots fehlschlagen. Verifizieren Sie, dass der Ziel-Snapshot den Status `AVAILABLE` hat, bevor Sie einen Quell-Snapshot löschen.
- Pro Konto können bis zu fünf Snapshot-Kopieranforderungen an eine einzelne Zielregion aktiv sein.
- Je nach den AWS-Regionen Beteiligten und der Menge der zu kopierenden Daten kann es Stunden dauern, bis eine regionsübergreifende Snapshot-Kopie abgeschlossen ist. In einigen Fällen kann es eine große Anzahl von regionsübergreifenden Snapshot-Kopieranforderungen aus einer bestimmten AWS-Region geben. In diesen Fällen kann Amazon DocumentDB neue

regionsübergreifende Kopieranforderungen von dieser Quelle AWS-Region in eine Warteschlange stellen, bis einige gerade ausgeführte Kopien abgeschlossen sind. Zu Kopieranforderungen, die sich in der Warteschlange befinden, werden keine Fortschrittsinformationen angezeigt. Fortschrittsinformationen werden angezeigt, sobald der Kopiervorgang gestartet wird.

Umgang mit Verschlüsselungen

Sie können einen Snapshot kopieren, der mit einem AWS KMS -Verschlüsselungsschlüssel verschlüsselt wurde. Wenn Sie einen verschlüsselten Snapshot kopieren, muss auch die Kopie des Snapshots verschlüsselt werden. Wenn Sie einen verschlüsselten Snapshot innerhalb desselben kopieren AWS-Region, können Sie die Kopie mit demselben AWS KMS Verschlüsselungsschlüssel wie den ursprünglichen Snapshot verschlüsseln, oder Sie können einen anderen Verschlüsselungsschlüssel angeben. AWS KMS Wenn Sie einen verschlüsselten Snapshot regionsübergreifend kopieren, können Sie für die Kopie nicht denselben AWS KMS Verschlüsselungsschlüssel verwenden, der für den Quell-Snapshot verwendet wurde, da dies bei AWS KMS Schlüsseln der Fall ist. Region-specific Stattdessen müssen Sie einen AWS KMS Schlüssel angeben, der im Ziel AWS-Region n gültig ist.

Der Quell-Snapshot bleibt den gesamten Kopiervorgang über verschlüsselt. Weitere Informationen finden Sie unter Datenschutz in Amazon DocumentDB.

> **Note**
>
> Bei Amazon DocumentDB-Cluster-Snapshots können Sie einen unverschlüsselten Cluster-Snapshot nicht verschlüsseln, wenn Sie den Snapshot kopieren.

Überlegungen zu Parametergruppen

Wenn Sie einen Snapshot regionsübergreifend kopieren, enthält die Kopie nicht die Parametergruppe, die vom ursprünglichen Amazon DocumentDB-Cluster verwendet wurde. Wenn Sie einen Snapshot wiederherstellen, um einen neuen Cluster zu erstellen, erhält dieser Cluster die Standardparametergruppe für den, in dem AWS-Region er erstellt wurde. Um dem neuen Cluster dieselben Parameter wie dem ursprünglichen Cluster zuzuweisen, müssen Sie wie folgt vorgehen:

1. Erstellen Sie im Ziel AWS-Region eine Amazon DocumentDB-Cluster-Parametergruppe mit denselben Einstellungen wie der ursprüngliche Cluster. Wenn im neuen bereits eine vorhanden ist AWS-Region, können Sie diese verwenden.

2. Nachdem Sie den Snapshot im Ziel wiederhergestellt haben AWS-Region, ändern Sie den neuen Amazon DocumentDB-Cluster und fügen Sie die neue oder vorhandene Parametergruppe aus dem vorherigen Schritt hinzu. Weitere Informationen finden Sie unter Ändern eines Amazon DocumentDB-Clusters.

Einen Cluster-Snapshot kopieren

Sie können einen Amazon DocumentDB-Cluster wie folgt mit AWS-Managementkonsole oder dem AWS CLI kopieren.

Using the AWS-Managementkonsole

Gehen Sie wie folgt vor, um mithilfe von eine Kopie eines Cluster-Snapshots zu erstellen. AWS-Managementkonsole Dieses Verfahren eignet sich für das Kopieren verschlüsselter oder unverschlüsselter Cluster-Snapshots in derselben Region AWS-Region oder regionsübergreifend.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Wählen Sie im Navigationsbereich Snapshots aus und klicken Sie dann auf die Schaltfläche links neben dem Snapshot, den Sie kopieren möchten.

   > **Tip**
   >
   > Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Wählen Sie im Menü Actions die Option Copy aus.
4. Füllen Sie auf der daraufhin angezeigten Seite „Kopie des Cluster-Snapshots erstellen“ den Abschnitt „Einstellungen“ aus.

   a. Zielregion — Optional. Um den Cluster-Snapshot in einen anderen zu kopieren AWS-Region, wählen Sie diesen AWS-Region als Zielregion aus.

   b. Neue Snapshot-ID — Geben Sie einen Namen für den neuen Snapshot ein.

      Einschränkungen bei der Benennung von Ziel-Snapshots:

      - Kann nicht der Namen eines vorhandenen Snapshots sein.

- Die Länge besteht aus [1—63] Buchstaben, Zahlen oder Bindestrichen.
- Muss mit einem Buchstaben beginnen.
- Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.
- Muss für alle Cluster in Amazon RDS, Neptune und Amazon DocumentDB pro Region eindeutig sein. AWS-Konto

c. Tags kopieren — Um alle Tags, die Sie in Ihrem Quell-Snapshot haben, in Ihre Snapshot-Kopie zu kopieren, wählen Sie Tags kopieren.

5. Füllen Sie den Encryption-at-rest Abschnitt aus.

a. Verschlüsselung im Ruhezustand — Wenn Ihr Snapshot nicht verschlüsselt ist, stehen Ihnen diese Optionen nicht zur Verfügung, da Sie aus einem unverschlüsselten Snapshot keine verschlüsselte Kopie erstellen können. Wenn Ihr Snapshot verschlüsselt ist, können Sie den Wert ändern, der bei der Verschlüsselung im Ruhezustand AWS KMS key verwendet wird.

Weitere Hinweise zum Verschlüsseln von Snapshot-Kopien finden Sie unterCluster-Snapshot-Verschlüsselung kopieren.

Weitere Informationen zur Verschlüsselung im Ruhezustand finden Sie unter Verschlüsselung von Amazon DocumentDB-Daten im Ruhezustand.

b. AWS KMS Schlüssel — Wählen Sie aus der Dropdownliste eine der folgenden Optionen aus:

- (Standard) aws/rds — Die Kontonummer und die AWS KMS Schlüssel-ID werden nach dieser Option aufgelistet.
- <some-key-name>— Wenn Sie einen Schlüssel erstellt haben, wird er aufgelistet und kann von Ihnen ausgewählt werden.
- Geben Sie einen Schlüssel-ARN ein — Geben Sie in das ARN-Feld den Amazon-Ressourcennamen (ARN) für Ihren AWS KMS Schlüssel ein. Das ARN-Format lautet `arn:aws:kms:<region>:<accountID>:key/<key-id>` .

6. Wenn Sie eine Kopie des ausgewählten Snapshots erstellen möchten, wählen Sie Copy Snapshot (Snapshot kopieren). Alternativ können Sie Abbrechen wählen, um keine Kopie des Snapshots zu erstellen.

Using the AWS CLI

Um eine Kopie eines unverschlüsselten Cluster-Snapshots mithilfe von zu erstellen AWS CLI, verwenden Sie den `copy-db-cluster-snapshot` Vorgang mit den folgenden Parametern. Wenn Sie den Snapshot in einen anderen kopieren AWS-Region, führen Sie den Befehl in dem aus, in den der Snapshot kopiert werden soll. AWS-Region

- **`--source-db-cluster-snapshot-identifier`** – Erforderlich. Die ID des zu kopierenden Cluster-Snapshots, der kopiert werden soll. Der Cluster-Snapshot muss vorhanden sein und muss sich im verfügbaren Zustand befindet. Wenn Sie den Snapshot in einen anderen kopieren AWS-Region oder wenn Sie einen gemeinsam genutzten Cluster-Snapshot kopieren, muss dieser Bezeichner im ARN-Format des Quell-Cluster-Snapshots vorliegen. Bei diesem Parameter wird nicht zwischen Groß- und Kleinschreibung unterschieden.
- **`--target-db-cluster-snapshot-identifier`** – Erforderlich. Die ID des neuen Cluster-Snapshots, der aus dem Quell-Cluster-Snapshot erstellt werden soll. Bei diesem Parameter wird nicht zwischen Groß- und Kleinschreibung unterschieden.

  Einschränkungen bei der Benennung von Ziel-Snapshots:

  - Kann nicht der Namen eines vorhandenen Snapshots sein.
  - Die Länge besteht aus [1—63] Buchstaben, Zahlen oder Bindestrichen.
  - Muss mit einem Buchstaben beginnen.
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.
  - Muss für alle Cluster in Amazon RDS, Neptune und Amazon DocumentDB pro Region eindeutig sein. AWS-Konto
- **`--source-region`**— Wenn Sie den Snapshot in einen anderen kopieren AWS-Region, geben Sie an, aus welchem der AWS-Region verschlüsselte Cluster-Snapshot kopiert werden soll.

  Wenn Sie den Snapshot in einen anderen kopieren AWS-Region und nichts angeben`--source-region`, müssen Sie stattdessen die `pre-signed-url` Option angeben. Der `pre-signed-url` Wert muss eine URL sein, die eine signierte Signature Version 4-Anfrage für den Aufruf der `CopyDBClusterSnapshot` Aktion in der Quelle AWS-Region enthält, aus der der Cluster-Snapshot kopiert wurde. Weitere Informationen zum finden Sie `pre-signed-url` unter CopyDBClusterSnapshot.
- **`--kms-key-id`**— Die KMS-Schlüssel-ID für den Schlüssel, der zum Verschlüsseln der Kopie des Cluster-Snapshots verwendet werden soll.

Wenn Sie einen verschlüsselten Cluster-Snapshot in einen anderen kopieren AWS-Region, ist dieser Parameter erforderlich. Sie müssen einen KMS-Schlüssel für das Ziel angeben AWS-Region.

Wenn Sie einen verschlüsselten Cluster-Snapshot in denselben kopieren AWS-Region, ist der AWS KMS Schlüsselparameter optional. Die Kopie des Cluster-Snapshots ist mit demselben AWS KMS Schlüssel wie der Quell-Cluster-Snapshot verschlüsselt. Wenn Sie einen neuen AWS KMS Verschlüsselungsschlüssel angeben möchten, der zum Verschlüsseln der Kopie verwendet werden soll, können Sie dies mit diesem Parameter tun.

- **`--copy-tags`**— Fakultativ. Die Tags und Werte, die kopiert werden sollen.

Um einen Kopiervorgang abzubrechen, sobald er gerade ausgeführt wird, können Sie den Ziel-Cluster-Snapshot löschen, der durch `--target-db-cluster-snapshot-identifier` oder `TargetDBClusterSnapshotIdentifier` während sich dieser Cluster-Snapshot im Kopierstatus befindet.

Example

Beispiel 1: Kopieren Sie einen unverschlüsselten Snapshot in dieselbe Region

Im folgenden AWS CLI Beispiel wird eine Kopie von `sample-cluster-snapshot` Named `sample-cluster-snapshot-copy` in erstellt, die dem Quell-Snapshot entspricht AWS-Region . Beim Erstellen der Kopie werden alle Tags des ursprünglichen Snapshots in die Snapshot-Kopie übernommen.

Für Linux, macOS oder Unix:

```
aws docdb copy-db-cluster-snapshot \
    --source-db-cluster-snapshot-identifier sample-cluster-snapshot \
    --target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy \
    --copy-tags
```

Für Windows:

```
aws docdb copy-db-cluster-snapshot ^
    --source-db-cluster-snapshot-identifier sample-cluster-snapshot ^
    --target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy ^
    --copy-tags
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
    "DBClusterSnapshot": {
        "AvailabilityZones": [
            "us-east-1a",
            "us-east-1b",
            "us-east-1c"
        ],
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot-copy",
        "DBClusterIdentifier": "sample-cluster",
        "SnapshotCreateTime": "2020-03-27T08:40:24.805Z",
        "Engine": "docdb",
        "Status": "copying",
        "Port": 0,
        "VpcId": "vpc-abcd0123",
        "ClusterCreateTime": "2020-01-10T22:13:38.261Z",
        "MasterUsername": "master-user",
        "EngineVersion": "4.0.0",
        "SnapshotType": "manual",
        "PercentProgress": 0,
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:111122223333:key/sample-key-id",
        "DBClusterSnapshotArn": "arn:aws:rds:us-east-1:111122223333:cluster-
snapshot:sample-cluster-snapshot-copy",
        "SourceDBClusterSnapshotArn": "arn:aws:rds:us-east-1:111122223333:cluster-
snapshot:sample-cluster-snapshot"
    }
}
```

Example

Beispiel 2: Kopieren Sie einen unverschlüsselten Snapshot AWS-Regionen

Im folgenden AWS CLI Beispiel wird eine Kopie von erstelltsample-cluster-snapshot, die den ARN `arn:aws:rds:us-east-1:123456789012:cluster-snapshot:sample-cluster-snapshot` enthält. Diese Kopie hat einen Namen `sample-cluster-snapshot-copy` und befindet sich in dem, AWS-Region in dem der Befehl ausgeführt wird.

Für Linux, macOS oder Unix:

```
aws docdb copy-db-cluster-snapshot \
```

```
    --source-db-cluster-snapshot-identifier arn:aws:rds:us-
east-1:123456789012:cluster-snapshot:sample-cluster-snapshot \
    --target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy
```

Für Windows:

```
aws docdb copy-db-cluster-snapshot ^
    --source-db-cluster-snapshot-identifier arn:aws:rds:us-
east-1:123456789012:cluster-snapshot:sample-cluster-snapshot ^
    --target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
    "DBClusterSnapshot": {
        "AvailabilityZones": [
            "us-east-1a",
            "us-east-1b",
            "us-east-1c"
        ],
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot-copy",
        "DBClusterIdentifier": "sample-cluster",
        "SnapshotCreateTime": "2020-04-29T16:45:51.239Z",
        "Engine": "docdb",
        "AllocatedStorage": 0,
        "Status": "copying",
        "Port": 0,
        "VpcId": "vpc-abc0123",
        "ClusterCreateTime": "2020-04-28T16:43:00.294Z",
        "MasterUsername": "master-user",
        "EngineVersion": "4.0.0",
        "LicenseModel": "docdb",
        "SnapshotType": "manual",
        "PercentProgress": 0,
        "StorageEncrypted": false,
        "DBClusterSnapshotArn": "arn:aws:rds:us-east-1:123456789012:cluster-
snapshot:sample-cluster-snapshot-copy",
        "SourceDBClusterSnapshotArn": "arn:aws:rds:us-east-1:123456789012:cluster-
snapshot:sample-cluster-snapshot",
    }
}
```

Example

Beispiel 3: Kopieren Sie einen verschlüsselten Snapshot AWS-Regionen

Im folgenden AWS CLI Beispiel wird eine Kopie `sample-cluster-snapshot` von Von der Region us-west-2 zur Region us-east-1 erstellt. Dieser Befehl wird in der Region us-east-1 aufgerufen.

Für Linux, macOS oder Unix:

```
aws docdb copy-db-cluster-snapshot \
    --source-db-cluster-snapshot-identifier arn:aws:rds:us-
west-2:123456789012:cluster-snapshot:sample-cluster-snapshot \
    --target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy \
    --source-region us-west-2 \
    --kms-key-id sample-us-east-1-key
```

Für Windows:

```
aws docdb copy-db-cluster-snapshot ^
    --source-db-cluster-snapshot-identifier arn:aws:rds:us-
west-2:123456789012:cluster-snapshot:sample-cluster-snapshot ^
    --target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy ^
    --source-region us-west-2 ^
    --kms-key-id sample-us-east-1-key
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
    "DBClusterSnapshot": {
        "AvailabilityZones": [],
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot-copy",
        "DBClusterIdentifier": "sample-cluster",
        "SnapshotCreateTime": "2020-04-29T16:45:53.159Z",
        "Engine": "docdb",
        "AllocatedStorage": 0,
        "Status": "copying",
        "Port": 0,
        "ClusterCreateTime": "2020-04-28T16:43:07.129Z",
        "MasterUsername": "chimera",
        "EngineVersion": "4.0.0",
        "LicenseModel": "docdb",
        "SnapshotType": "manual",
```

```
        "PercentProgress": 0,
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/0846496a-
d48e-41c4-9353-86d8301d7e35",
        "DBClusterSnapshotArn": "arn:aws:rds:us-east-1:123456789012:cluster-
snapshot:sample-cluster-snapshot-copy",
        "SourceDBClusterSnapshotArn": "arn:aws:rds:us-west-2:123456789012:cluster-
snapshot:sample-cluster-snapshot",
    }
}
```

Example

Beispiel 4: Kopieren Sie einen unverschlüsselten gemeinsamen Snapshot AWS-Regionen

Das folgende AWS CLI Beispiel, account -123456789012, erstellt eine Kopie eines unverschlüsselten Cluster-Snapshots, der von Account - 999999999999 von der Region us-east-1 zur Region us-west-2 `sample-cluster-snapshot` gemeinsam genutzt wird. Dieser Befehl wird in der Region us-west-2 aufgerufen. Weitere Informationen zum Freigeben von Snapshots finden Sie unter Freigeben eines Snapshots.

Für Linux, macOS oder Unix:

```
aws docdb copy-db-cluster-snapshot \
--region us-west-2 \
--source-db-cluster-snapshot-identifier arn:aws:rds:us-east-1:999999999999:cluster-
snapshot:sample-cluster-snapshot \
--target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy
```

Für Windows:

```
aws docdb copy-db-cluster-snapshot ^
--region us-west-2 ^
--source-db-cluster-snapshot-identifier arn:aws:rds:us-east-1:999999999999:cluster-
snapshot:sample-cluster-snapshot ^
--target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
 "DBClusterSnapshots": [
 {
```

```
 "AvailabilityZones": [],
 "DBClusterSnapshotIdentifier": "sample-cluster-snapshot-copy",
 "DBClusterIdentifier": "sample-cluster",
 "SnapshotCreateTime": "2025-08-22T11:27:00.497000+00:00",
 "Engine": "docdb",
 "Status": "copying",
 "Port": 0,
 "ClusterCreateTime": "2024-07-02T16:44:50.246000+00:00",
 "MasterUsername": "master-user",
 "EngineVersion": "5.0.0",
 "SnapshotType": "manual",
 "PercentProgress": 0,
 "StorageEncrypted": false,
 "DBClusterSnapshotArn": "arn:aws:rds:us-west-2:123456789012:cluster-
snapshot:sample-cluster-snapshot-copy",
 "SourceDBClusterSnapshotArn": "arn:aws:rds:us-east-1:999999999999:cluster-
snapshot:sample-cluster-snapshot"
 }
 ]
}
```

Example

Beispiel 5: Kopieren Sie einen verschlüsselten gemeinsamen Snapshot AWS-Regionen

Das folgende AWS CLI Beispiel, account — `123456789012` erstellt eine Kopie eines verschlüsselten Cluster-Snapshots, der von einem Konto `sample-cluster-snapshot` gemeinsam genutzt wird — `999999999999` von der Region us-east-1 in die Region us-west-2. Der Ziel-Snapshot ist mit einem vom Kunden verwalteten KMS-Schlüssel verschlüsselt -. `arn:aws:kms:us-west-2:123456789012:key/6c1f3264-1797-472b-ba37-03011e682d28` Dieser Befehl wird in der Region us-west-2 aufgerufen. Weitere Informationen zum Freigeben von Snapshots finden Sie unter Freigeben eines Snapshots.

Für Linux, macOS oder Unix:

```
aws docdb copy-db-cluster-snapshot \
--region us-west-2 \
--source-db-cluster-snapshot-identifier arn:aws:rds:us-east-1:999999999999:cluster-
snapshot:sample-cluster-snapshot \
--target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy \
--kms-key-id arn:aws:kms:us-west-2:123456789012:key/6c1f3264-1797-472b-
ba37-03011e682d28
```

Für Windows:

```
aws docdb copy-db-cluster-snapshot ^
--region us-west-2 ^
--source-db-cluster-snapshot-identifier arn:aws:rds:us-east-1:999999999999:cluster-
snapshot:sample-cluster-snapshot ^
--target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy ^
--kms-key-id arn:aws:kms:us-west-2:123456789012:key/6c1f3264-1797-472b-
ba37-03011e682d28
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
 "DBClusterSnapshots": [
 {
 "AvailabilityZones": [],
 "DBClusterSnapshotIdentifier": "sample-cluster-snapshot-copy",
 "DBClusterIdentifier": "sample-cluster",
 "SnapshotCreateTime": "2025-08-22T11:27:00.497000+00:00",
 "Engine": "docdb",
 "Status": "copying",
 "Port": 0,
 "ClusterCreateTime": "2024-07-02T16:44:50.246000+00:00",
 "MasterUsername": "master-user",
 "EngineVersion": "5.0.0",
 "SnapshotType": "manual",
 "PercentProgress": 0,
 "StorageEncrypted": true,
 "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/6c1f3264-1797-472b-
ba37-03011e682d28",
 "DBClusterSnapshotArn": "arn:aws:rds:us-west-2:123456789012:cluster-
snapshot:sample-cluster-snapshot-copy",
 "SourceDBClusterSnapshotArn": "arn:aws:rds:us-east-1:999999999999:cluster-
snapshot:sample-cluster-snapshot"
 }
 ]
}
```

> **Note**
>
> Weitere Hinweise zum Verschlüsseln von Snapshot-Kopien finden Sie unter. Cluster-Snapshot-Verschlüsselung kopieren

Weitere Informationen zur Verschlüsselung im Ruhezustand finden Sie unter Verschlüsselung von Amazon DocumentDB-Daten im Ruhezustand.

Amazon DocumentDB-Cluster-Snapshots teilen

Mithilfe von Amazon DocumentDB können Sie einen manuellen Cluster-Snapshot auf folgende Weise teilen:

- Durch das Teilen eines manuellen Cluster-Snapshots, ob verschlüsselt oder unverschlüsselt, können autorisierte AWS Konten den Snapshot kopieren.
- Durch die gemeinsame Nutzung eines manuellen Cluster-Snapshots, ob verschlüsselt oder unverschlüsselt, können autorisierte AWS Konten einen Cluster direkt aus dem Snapshot wiederherstellen, anstatt eine Kopie davon zu erstellen und von dort aus wiederherzustellen.

> **Note**
>
> Um einen automatisierten Cluster-Snapshot gemeinsam zu nutzen, erstellen Sie einen manuellen Cluster-Snapshot, indem Sie den automatisierten Snapshot kopieren und diese Kopie dann gemeinsam nutzen. Dieser Vorgang gilt auch für durch AWS Backups generierte Ressourcen.

Sie können einen manuellen Snapshot mit bis zu 20 anderen teilen. AWS-Konten Sie können darüber hinaus einen nicht verschlüsselten Snapshot als öffentlich freigeben. Damit ist der Snapshot für alle -Konten verfügbar. Vergewissern Sie sich bei der Freigabe eines Snapshots als öffentlich darauf, dass in Ihren öffentlichen Snapshots keine privaten Informationen enthalten sind.

Wenn Sie manuelle Snapshots mit anderen AWS-Konten teilen und Sie einen Cluster mithilfe der AWS CLI oder der Amazon DocumentDB-API aus einem gemeinsam genutzten Snapshot wiederherstellen, müssen Sie den Amazon-Ressourcennamen (ARN) des gemeinsam genutzten Snapshots als Snapshot-ID angeben.

Freigeben eines verschlüsselten Snapshots

Die folgenden Einschränkungen gelten für die Freigabe verschlüsselter Snapshots:

- Sie können verschlüsselte Snapshots nicht als öffentlich freigeben.

- Sie können keinen Snapshot teilen, der mit dem AWS KMS Standardverschlüsselungsschlüssel des Kontos verschlüsselt wurde, das den Snapshot geteilt hat.

Führen Sie die folgenden Schritte aus, um verschlüsselte Snapshots freizugeben.

1. Teilen Sie den Verschlüsselungsschlüssel AWS Key Management Service (AWS KMS), der zum Verschlüsseln des Snapshots verwendet wurde, mit allen Konten, die auf den Snapshot zugreifen sollen.

   Sie können AWS KMS Verschlüsselungsschlüssel mit anderen AWS Konten teilen, indem Sie die anderen Konten zur AWS KMS Schlüsselrichtlinie hinzufügen. Einzelheiten zum Aktualisieren einer Schlüsselrichtlinie finden Sie im AWS Key Management Service Entwicklerhandbuch unter Verwenden von Schlüsselrichtlinien in AWS KMS. Ein Beispiel für die Erstellung einer Schlüsselrichtlinie finden Sie unter Erstellen einer IAM-Richtlinie, um das Kopieren des verschlüsselten Snapshots zu erlauben an späterer Stelle in diesem Thema.
2. Verwenden Sie AWS CLI, wie unten gezeigt, um den verschlüsselten Snapshot mit den anderen Konten zu teilen.

Ermöglichen Sie den Zugriff auf eine AWS KMS Verschlüsselungsschlüssel

Damit ein anderer AWS-Konto Benutzer einen verschlüsselten Snapshot kopieren kann, der von Ihrem Konto aus geteilt wurde, muss das Konto, mit dem Sie Ihren Snapshot teilen, Zugriff auf den AWS KMS Schlüssel haben, mit dem der Snapshot verschlüsselt wurde. Um einem anderen Konto den Zugriff auf einen AWS KMS Schlüssel zu ermöglichen, aktualisieren Sie die Schlüsselrichtlinie für den AWS KMS Schlüssel mit dem ARN des Kontos, für das Sie als Principal in der AWS KMS Schlüsselrichtlinie den Schlüssel verwenden. Lassen Sie anschließend die Aktion `kms:CreateGrant` zu.

Nachdem Sie einem Konto Zugriff auf Ihren AWS KMS Verschlüsselungsschlüssel gewährt haben, muss dieses Konto einen AWS Identity and Access Management (IAM-) Benutzer erstellen, um Ihren verschlüsselten Snapshot zu kopieren, sofern es noch keinen hat. Darüber hinaus muss dieses Konto diesem IAM-Benutzer eine IAM-Richtlinie zuordnen, die es dem Benutzer ermöglicht, einen verschlüsselten Snapshot mithilfe Ihres Schlüssels zu kopieren. AWS KMS Das Konto muss ein IAM-Benutzer sein und kann aufgrund von Sicherheitsbeschränkungen keine AWS-Konto Root-Identität sein. AWS KMS

Im folgenden Beispiel für eine Schlüsselrichtlinie ist der Benutzer 123451234512 der Besitzer des Verschlüsselungsschlüssels. AWS KMS Bei Benutzer 123456789012 handelt es sich um das Konto, für das der Schlüssel freigegeben wird. Diese aktualisierte Schlüsselrichtlinie gewährt dem Konto Zugriff auf den Schlüssel. AWS KMS Dazu wird der ARN für die AWS-Konto Stammidentität des Benutzers 123456789012 als Prinzipal für die Richtlinie angegeben und die Aktion zugelassen. `kms:CreateGrant`

JSON

```
{
    "Id": "key-policy-1",
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {"AWS": [
                "arn:aws:iam::123451234512:user/KeyUser",
                "arn:aws:iam::123456789012:root"
            ]},
            "Action": [
                "kms:CreateGrant",
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"},
            {
            "Sid": "Allow attachment of persistent resources",
            "Effect": "Allow",
            "Principal": {"AWS": [
                "arn:aws:iam::123451234512:user/KeyUser",
                "arn:aws:iam::123456789012:root"
            ]},
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": "*",
```

```
            "Condition": {"Bool": {"kms:GrantIsForAWSResource": true}}
        }
    ]
}
```

Erstellen einer IAM-Richtlinie, um das Kopieren des verschlüsselten Snapshots zu erlauben

Wenn die externe Person Zugriff auf Ihren AWS KMS Schlüssel AWS-Konto hat, kann der Besitzer dieses Kontos eine Richtlinie erstellen, die es einem für das Konto erstellten IAM-Benutzer ermöglicht, einen verschlüsselten Snapshot zu kopieren, der mit diesem Schlüssel verschlüsselt ist. AWS KMS

Das folgende Beispiel zeigt eine Richtlinie, die an einen IAM-Benutzer für AWS-Konto 123456789012 angehängt werden kann. Die Richtlinie ermöglicht es dem IAM-Benutzer, einen gemeinsam genutzten Snapshot vom Konto 123451234512 zu kopieren, der mit dem Schlüssel in der Region us-west-2 verschlüsselt wurde. AWS KMS `c989c1dd-a3f2-4a5d-8d96-e793d082ab26`

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowUseOfTheKey",
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey",
                "kms:CreateGrant",
                "kms:RetireGrant"
            ],
            "Resource": ["arn:aws:kms:us-west-2:123451234512:key/c989c1dd-
a3f2-4a5d-8d96-e793d082ab26"]
        },
        {
            "Sid": "AllowAttachmentOfPersistentResources",
```

```
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": ["arn:aws:kms:us-west-2:123451234512:key/c989c1dd-
a3f2-4a5d-8d96-e793d082ab26"],
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": true
                }
            }
        }
    ]
}
```

Einzelheiten zum Aktualisieren einer Schlüsselrichtlinie finden Sie im Entwicklerhandbuch unter Verwenden wichtiger Richtlinien. AWS KMS AWS Key Management Service

Freigeben eines Snapshots

Sie können einen manuellen Amazon DocumentDB-Cluster-Snapshot (oder eine Kopie eines automatisierten Snapshots) teilen, indem Sie den AWS-Managementkonsole oder den AWS CLI:

Using the AWS-Managementkonsole

Gehen Sie wie folgt vor AWS-Managementkonsole, um einen Snapshot mithilfe von zu teilen:

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Wählen Sie im Navigationsbereich die Option Snapshots.
3. Wählen Sie den manuellen Snapshot, den Sie freigeben möchten.
4. Wählen Sie im Dropdownmenü Aktionen die Option Teilen aus.
5. Wählen Sie eine der folgenden Optionen für die Sichtbarkeit von DB-Snapshots:

   - Wenn die Quelle unverschlüsselt ist, wählen Sie Öffentlich, damit alle AWS Konten einen Cluster aus Ihrem manuellen Snapshot wiederherstellen können. Oder wählen Sie Privat, um nur AWS Konten zu erlauben, die Sie angeben, einen Cluster aus Ihrem manuellen Snapshot wiederherzustellen.

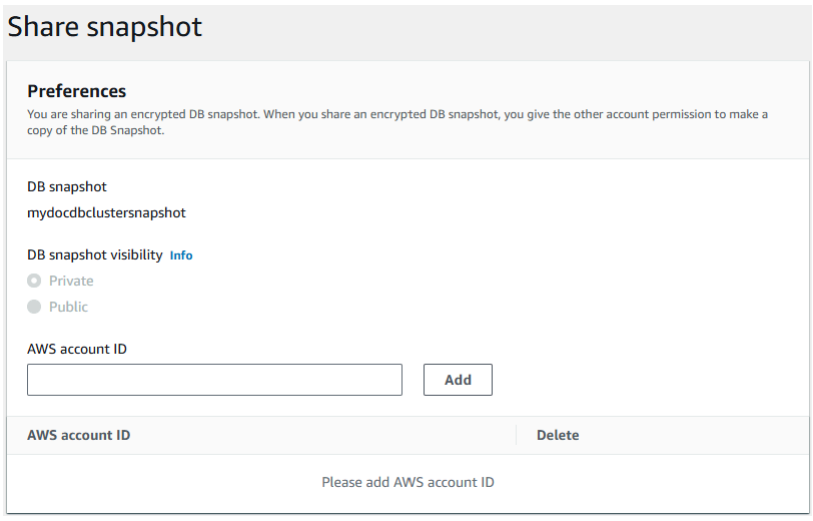
> **Warning**
>
> Wenn Sie die Sichtbarkeit von DB-Snapshots auf Öffentlich setzen, können alle AWS Konten einen Cluster aus Ihrem manuellen Snapshot wiederherstellen und haben Zugriff auf Ihre Daten. Geben Sie keine manuellen Cluster-Snapshots, die private Informationen enthalten, als öffentlich frei.

- Ist die Quelle verschlüsselt, ist DB Snapshot Visibility (Sichtbarkeit des DB-Snapshots) auf Private (Privat) festgelegt, da verschlüsselte Snapshots nicht als öffentlich freigegeben werden können.

> **Note**
>
> Snapshots, die mit der Standardverschlüsselung verschlüsselt wurden, AWS KMS key können nicht geteilt werden.

6. Geben Sie AWS unter Konto-ID die AWS Konto-ID für ein Konto ein, dem Sie die Wiederherstellung eines Clusters aus Ihrem manuellen Snapshot gestatten möchten, und wählen Sie dann Hinzufügen aus. Wiederholen Sie diesen Vorgang, um weitere AWS Kontokennungen hinzuzufügen, bis zu 20 AWS Konten.

   Wenn Ihnen beim Hinzufügen einer AWS Konto-ID zur Liste der zulässigen Konten ein Fehler unterläuft, können Sie sie aus der Liste löschen, indem Sie rechts neben der falschen AWS Konto-ID auf Löschen klicken.



7. Nachdem Sie Kennungen für alle AWS Konten hinzugefügt haben, denen Sie die Wiederherstellung des manuellen Snapshots gestatten möchten, wählen Sie Speichern, um Ihre Änderungen zu speichern.

Using the AWS CLI

Um einen Snapshot mithilfe von zu teilen AWS CLI, verwenden Sie den Amazon DocumentDB-Vorgang`modify-db-snapshot-attribute`. Verwenden Sie den `--values-to-add` Parameter, um eine Liste der IDs für die hinzuzufügen AWS-Konten , die zur Wiederherstellung des manuellen Snapshots autorisiert sind.

Im folgenden Beispiel sind zwei AWS-Konto Bezeichner, 123451234512 und 123456789012, zulässig, um den benannten Snapshot wiederherzustellen. `manual-snapshot1` Außerdem wird der `all`-Attributwert entfernt, um den Snapshot als privat zu markieren.

Für Linux, macOS oder Unix:

```
aws docdb modify-db-cluster-snapshot-attribute \
    --db-cluster-snapshot-identifier sample-cluster-snapshot \
    --attribute-name restore \
    --values-to-add '["123451234512","123456789012"]'
```

Für Windows:

```
aws docdb modify-db-cluster-snapshot-attribute ^
    --db-cluster-snapshot-identifier sample-cluster-snapshot ^
    --attribute-name restore ^
    --values-to-add '["123451234512","123456789012"]'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
    "DBClusterSnapshotAttributesResult": {
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot",
        "DBClusterSnapshotAttributes": [
            {
                "AttributeName": "restore",
                "AttributeValues": [
                    "123451234512",
```

```
                    "123456789012"
                ]
            }
        ]
    }
}
```

Verwenden Sie den Parameter, um einen AWS-Konto Bezeichner aus der Liste zu entfernen. `--values-to-remove` Das folgende Beispiel verhindert, dass die AWS-Konto ID 123456789012 den Snapshot wiederherstellt.

Für Linux, macOS oder Unix:

```
aws docdb modify-db-cluster-snapshot-attribute \
    --db-cluster-snapshot-identifier sample-cluster-snapshot \
    --attribute-name restore \
    --values-to-remove '["123456789012"]'
```

Für Windows:

```
aws docdb modify-db-cluster-snapshot-attribute ^
    --db-cluster-snapshot-identifier sample-cluster-snapshot ^
    --attribute-name restore ^
    --values-to-remove '["123456789012"]'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
    "DBClusterSnapshotAttributesResult": {
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot",
        "DBClusterSnapshotAttributes": [
            {
                "AttributeName": "restore",
                "AttributeValues": [
                    "123451234512"
                ]
            }
        ]
    }
}
```

Wiederherstellung aus einem Cluster-Snapshot

Amazon DocumentDB (mit MongoDB-Kompatibilität) erstellt einen Cluster-Snapshot Ihres Speichervolumes. Sie können einen neuen Cluster erstellen, indem Sie ihn aus einem Cluster-Snapshot wiederherstellen. Wenn Sie den Cluster wiederherstellen, geben Sie den Namen des wiederherzustellenden Cluster-Snapshots und einen Namen für den neuen Cluster an, der durch die Wiederherstellung erstellt wird. Sie können nicht von einem Snapshot auf einen bestehenden Cluster wiederherstellen, da bei der Wiederherstellung ein neuer Cluster erstellt wird.

Wenn Sie einen Cluster aus einem Cluster-Snapshot wiederherstellen:

- Diese Aktion stellt nur den Cluster wieder her, nicht die Instances für diesen Cluster. Sie müssen die Aktion `create-db-instance` aufrufen, um Instances für den wiederhergestellten Cluster zu erstellen, wobei Sie in `--db-cluster-identifier` die ID des wiederhergestellten Clusters angeben. Sie können Instances erst erstellen, nachdem der Cluster den Status available hat.
- Sie können einen verschlüsselten Snapshot nicht in einem unverschlüsselten Cluster wiederherstellen. Sie können jedoch einen unverschlüsselten Snapshot in einem verschlüsselten Cluster wiederherstellen, indem Sie den Schlüssel angeben. AWS KMS
- Um einen Cluster aus einem verschlüsselten Snapshot wiederherzustellen, müssen Sie Zugriff auf den AWS KMS Schlüssel haben.

> **Note**
>
> Sie können einen 3.6-Cluster nicht auf einen 4.0-Cluster wiederherstellen, aber Sie können von einer Cluster-Version zu einer anderen migrieren. Weitere Informationen finden Sie unter Migration und Aktualisierung von Amazon DocumentDB.

Using the AWS-Managementkonsole

Das folgende Verfahren zeigt, wie Sie mithilfe der Amazon DocumentDB-Managementkonsole einen Amazon DocumentDB-Cluster aus einem Cluster-Snapshot wiederherstellen.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Wählen Sie im Navigationsbereich Snapshots und dann die Schaltfläche links neben dem Snapshot aus, mit der Sie einen Cluster wiederherstellen möchten.

> **Tip**
>
> Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Wählen Sie im Menü Aktionen die Option Restore (Wiederherstellen).
4. Füllen Sie auf der Seite Restore snapshot (Snapshot wiederherstellen) den Abschnitt Configuration (Konfiguration) aus.

   a. Cluster-ID — Der Name für den neuen Cluster. Sie können den von Amazon DocumentDB bereitgestellten Namen akzeptieren oder einen von Ihnen bevorzugten Namen eingeben. Der von Amazon DocumentDB bereitgestellte Name hat das Format `docdb-` plus einen UTC-Zeitstempel, z. B. `docdb-yyyy-mm-dd-hh-mm-ss`

   b. Instanzklasse — Die Instanzklasse für den neuen Cluster. Sie können die Standard-Instance-Klasse akzeptieren oder eine Instance-Klasse aus der Dropdown-Liste auswählen.

   c. Anzahl der Instanzen — Die Anzahl der Instanzen, die Sie mit diesem Cluster erstellen möchten. Sie können die Standardeinstellung von 3 Instanzen (1 primäre read/write und 2 schreibgeschützte Replikate) akzeptieren oder die Anzahl der Instanzen aus der Dropdownliste auswählen.
5. Wählen Sie für die Cluster-Speicherkonfiguration eine Speicheroption.

> **Note**
>
> Die Amazon I/O-Optimized DocumentDB-Speicherkonfiguration ist nur in den Amazon DocumentDB-Engine-Versionen 5.0 und 8.0 verfügbar.

6. Wenn Sie mit der Cluster-Konfiguration zufrieden sind, wählen Sie Restore cluster (Cluster wiederherstellen) aus und warten Sie, bis Ihr Cluster wiederhergestellt ist.
7. Wenn Sie es vorziehen, einige Konfigurationen zu ändern, z. B. eine nicht standardmäßige Amazon VPC oder Sicherheitsgruppe anzugeben, wählen Sie unten links auf der Seite die Option Erweiterte Einstellungen anzeigen aus und fahren Sie dann mit den folgenden Schritten fort.

   a. Vervollständigen Sie den Bereich Network settings (Netzwerkeinstellungen).

- Virtual Private Cloud (VPC) — Akzeptieren Sie die aktuelle VPC oder wählen Sie eine VPC aus der Dropdownliste aus.
- Subnetzgruppe — Akzeptieren Sie die `default` Subnetzgruppe oder wählen Sie eine aus der Dropdownliste aus.
- VPC-Sicherheitsgruppen — Akzeptieren Sie die `default (VPC)` Sicherheitsgruppe oder wählen Sie eine aus der Liste aus.

b. Vervollständigen Sie den Abschnitt Cluster options (Cluster-Optionen).

- Datenbankport — Übernehmen Sie den Standardport`27017`, oder verwenden Sie den Aufwärts- oder Abwärtspfeil, um den Port festzulegen, den Sie für Anwendungsverbindungen verwenden möchten.

c. Vervollständigen Sie den Abschnitt Encryption (Verschlüsselung).

- Verschlüsselung im Ruhezustand — Wenn Ihr Snapshot verschlüsselt ist, stehen Ihnen diese Optionen nicht zur Verfügung. Wenn er nicht verschlüsselt ist, können Sie eine der folgenden Optionen auswählen:
  - Um alle Daten Ihres Clusters zu verschlüsseln, wählen Sie Enable encryption-at-rest (Verschlüsselung im Ruhezustand aktivieren) aus. Wenn Sie diese Option wählen, müssen Sie einen KMS-Schlüssel angeben.
  - Um die Daten Ihres Clusters nicht zu verschlüsseln, wählen Sie Disable encryption-at-rest (Verschlüsselung im Ruhezustand deaktivieren) aus. Wenn Sie diese Option wählen, sind Sie mit dem Abschnitt "Verschlüsselung" fertig.
- AWS KMS Schlüssel — Wählen Sie eine der folgenden Optionen aus der Dropdownliste aus:
  - (Standard) aws/rds — Die Kontonummer und die AWS KMS Schlüssel-ID werden nach dieser Option aufgelistet.
  - Customer-managed Schlüssel — Diese Option ist nur verfügbar, wenn Sie in der AWS Identity and Access Management (IAM) -Konsole einen IAM-Verschlüsselungsschlüssel erstellt haben. Sie können den Schlüssel für die Verschlüsselung Ihres Clusters auswählen.
  - Geben Sie einen Schlüssel-ARN ein — Geben Sie im ARN-Feld den Amazon-Ressourcennamen (ARN) für Ihren AWS KMS Schlüssel ein. Das ARN-Format lautet `arn:aws:kms:<region>:<accountID>:key/<key-id>`.

d. Füllen sie den Abschnitt Log exports (Protokollexporte) aus.

- Wählen Sie die Protokolltypen aus, in denen Sie veröffentlichen möchten CloudWatch — Wählen Sie eine der folgenden Optionen:
  - Aktiviert — Ermöglicht Ihrem Cluster, DDL-Logging nach Amazon CloudWatch Logs zu exportieren.
  - Deaktiviert — Verhindert, dass Ihr Cluster DDL-Protokolle nach Amazon CloudWatch Logs exportiert. Disabled (Deaktiviert) ist die Voreinstellung.
- IAM-Rolle — Wählen Sie aus der Liste die Option RDS Service Linked Role aus.

e. Füllen Sie den Abschnitt Tags aus.

- Tag hinzufügen — Geben Sie im Schlüsselfeld den Namen für das Tag für Ihren Cluster ein. Geben Sie optional im Feld Value (Wert) den Tag-Wert ein. Tags werden zusammen mit AWS Identity and Access Management (IAM-) Richtlinien verwendet, um den Zugriff auf Amazon DocumentDB-Ressourcen zu verwalten und zu steuern, welche Aktionen auf die Ressourcen angewendet werden können.

f. Füllen Sie den Abschnitt Deletion protection (Löschschutz) aus.

- Löschschutz aktivieren — Schützt den Cluster vor versehentlichem Löschen. Wenn diese Option aktiviert ist, können Sie den Cluster nicht löschen.

8. Wählen Sie Restore cluster (Cluster wiederherstellen) aus.

Using the AWS CLI

Um einen Cluster aus einem Snapshot mithilfe von wiederherzustellen AWS CLI, verwenden Sie den `restore-db-cluster-from-snapshot` Vorgang mit den folgenden Parametern. Weitere Informationen finden Sie unter RestoreDBClusterFromSnapshot.

- **`--db-cluster-identifier`** – Erforderlich. Der Name des Clusters, der von der Operation erstellt wird. Ein Cluster mit diesem Namen darf vor dieser Operation nicht vorhanden sein.

  Einschränkungen bei der Benennung von Clustern:

  - Die Länge besteht aus [1—63] Buchstaben, Zahlen oder Bindestrichen.
  - Muss mit einem Buchstaben beginnen.
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.

  - Muss für alle Cluster in Amazon RDS, Neptune und Amazon DocumentDB pro Region eindeutig sein. AWS-Konto
- **`--snapshot-identifier`** – Erforderlich. Der Name des Snapshots, der für die Wiederherstellung verwendet wird. Ein Snapshot mit diesem Namen muss vorhanden sein und sich im verfügbaren Zustand befindet.
- **`--engine`** – Erforderlich. Der Wert muss `docdb` sein.
- **`--storage-type standard | iopt1`**— Fakultativ. Standard: `standard`.
- **`--kms-key-id`**— Fakultativ. Der ARN der AWS KMS Schlüssel-ID, die beim Wiederherstellen eines verschlüsselten Snapshots oder beim Verschlüsseln eines Clusters bei der Wiederherstellung aus einem unverschlüsselten Snapshot verwendet werden soll. Die Angabe der AWS KMS Schlüssel-ID führt dazu, dass der wiederhergestellte Cluster mit dem AWS KMS Schlüssel verschlüsselt wird, unabhängig davon, ob der Snapshot verschlüsselt war oder nicht.

  Das Format der `--kms-key-id` ist `arn:aws:kms:<region>:<accountID>:key/<key-id>`. Wenn Sie keinen Wert für den `--kms-key-id`-Parameter angeben, geschieht folgendes:

  - Wenn der `--snapshot-identifier` Snapshot-Eingang verschlüsselt ist, wird der wiederhergestellte Cluster mit demselben AWS KMS Schlüssel verschlüsselt, der zum Verschlüsseln des Snapshots verwendet wurde.
  - Wenn der Snapshot in `--snapshot-identifier` nicht verschlüsselt ist, dann ist der wiederhergestellte Cluster nicht verschlüsselt.

Für Linux, macOS oder Unix:

```
aws docdb restore-db-cluster-from-snapshot \
    --db-cluster-identifier sample-cluster-restore \
    --snapshot-identifier sample-cluster-snapshot \
    --engine docdb \
    --kms-key-id arn:aws:kms:us-east-1:123456789012:key/SAMPLE-KMS-KEY-ID
```

Für Windows:

```
aws docdb restore-db-cluster-from-snapshot ^
    --db-cluster-identifier sample-cluster-restore ^
    --snapshot-identifier sample-cluster-snapshot ^
    --engine docdb ^
    --kms-key-id arn:aws:kms:us-east-1:123456789012:key/SAMPLE-KMS-KEY-ID
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
    "DBCluster": {
        "AvailabilityZones": [
            "us-east-1c",
            "us-east-1b",
            "us-east-1a"
        ],
        "BackupRetentionPeriod": 1,
        "DBClusterIdentifier": "sample-cluster-restore",
        "DBClusterParameterGroup": "default.docdb4.0",
        "DBSubnetGroup": "default",
        "Status": "creating",
        "Endpoint": "sample-cluster-restore.cluster-node.us-
east-1.docdb.amazonaws.com",
        "ReaderEndpoint": "sample-cluster-restore.cluster-node.us-
east-1.docdb.amazonaws.com",
        "MultiAZ": false,
        "Engine": "docdb",
        "EngineVersion": "4.0.0",
        "Port": 27017,
        "MasterUsername": "<master-user>",
        "PreferredBackupWindow": "02:00-02:30",
        "PreferredMaintenanceWindow": "tue:09:50-tue:10:20",
        "DBClusterMembers": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abcdefgh",
                "Status": "active"
            }
        ],
        "HostedZoneId": "ABCDEFGHIJKLM",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/<sample-key-id>",
        "DbClusterResourceId": "cluster-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
        "DBClusterArn": "arn:aws:rds:us-east-1:<accountID>:cluster:sample-cluster-
restore",
        "AssociatedRoles": [],
        "ClusterCreateTime": "2020-04-01T01:43:40.871Z",
        "DeletionProtection": true
    }
}
```

Nachdem der Clusterstatus available lautet, erstellen Sie mindestens eine Instance für den Cluster.

Für Linux, macOS oder Unix:

```
aws docdb create-db-instance \
    --db-cluster-identifier sample-cluster-restore  \
    --db-instance-identifier sample-cluster-restore-instance \
    --availability-zone us-east-1b \
    --promotion-tier 2 \
    --db-instance-class db.r5.large \
    --engine docdb
```

Für Windows:

```
aws docdb create-db-instance ^
    --db-cluster-identifier sample-cluster-restore  ^
    --db-instance-identifier sample-cluster-restore-instance ^
    --availability-zone us-east-1b ^
    --promotion-tier 2 ^
    --db-instance-class db.r5.large ^
    --engine docdb
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "sample-cluster-restore-instance",
        "DBInstanceClass": "db.r5.large",
        "Engine": "docdb",
        "DBInstanceStatus": "creating",
        "PreferredBackupWindow": "02:00-02:30",
        "BackupRetentionPeriod": 1,
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abcdefgh",
                "Status": "active"
            }
        ],
        "AvailabilityZone": "us-west-2b",
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
```

```
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-6242c31a",
            "SubnetGroupStatus": "Complete",
            "Subnets": [
                {
                    "SubnetIdentifier": "subnet-abcdefgh",
                    "SubnetAvailabilityZone": {
                        "Name": "us-west-2a"
                    },
                    "SubnetStatus": "Active"
                },
                {
                    ...
                }
            ]
        },
        "PreferredMaintenanceWindow": "fri:09:43-fri:10:13",
        "PendingModifiedValues": {},
        "EngineVersion": "4.0.0",
        "AutoMinorVersionUpgrade": true,
        "PubliclyAccessible": false,
        "DBClusterIdentifier": "sample-cluster-restore",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/<sample-key-id>",
        "DbiResourceId": "db-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
        "CACertificateIdentifier": "rds-ca-2019",
        "PromotionTier": 2,
        "DBInstanceArn": "arn:aws:rds:us-east-1:<accountID>:db:sample-cluster-
restore-instance"
    }
}
```

Wiederherstellung auf einen bestimmten Zeitpunkt

Sie können einen Cluster zu einem beliebigen Zeitpunkt innerhalb des Aufbewahrungszeitraums für Backups des Clusters wiederherstellen, indem Sie das AWS-Managementkonsole oder AWS Command Line Interface (AWS CLI) verwenden.

> **Note**
>
> Sie können keine Point-in-Time-Wiederherstellung eines 3.6-Clusters auf einen 4.0-Cluster durchführen, aber Sie können von einer Clusterversion auf eine andere migrieren. Weitere Informationen finden Sie unter Migration und Aktualisierung von Amazon DocumentDB.

Beachten Sie Folgendes, wenn Sie einen Cluster zu einem bestimmten Zeitpunkt wiederherstellen.

- Der neue Cluster wird mit der gleichen Konfiguration wie der Quell-Cluster erstellt, nur dass der neue Cluster mit der Standard-Parametergruppe erstellt wird. Um die Parametergruppe des neuen Clusters auf die Parametergruppe des Quellclusters festzulegen, ändern Sie den Cluster, nachdem er den Status available (verfügbar) hat. Weitere Informationen zum Ändern eines Clusters finden Sie unter Ändern eines Amazon DocumentDB-Clusters.

Using the AWS-Managementkonsole

Sie können einen Cluster innerhalb des Aufbewahrungszeitraums für Backups zu einem bestimmten Zeitpunkt wiederherstellen, indem Sie in der AWS-Managementkonsole Folgendes ausführen.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Klicken Sie im Navigationsbereich auf Cluster. Wählen Sie in der Liste der Cluster die Schaltfläche links neben dem Cluster, den Sie wiederherstellen möchten.

   > **Tip**
   >
   > Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Wählen Sie im Menü Actions (Aktionen) die Option Restore to point in time (Zu einem bestimmten Zeitpunkt wiederherstellen).
4. Geben Sie im Bereich Restore time (Wiederherstellungszeit) Datum und Uhrzeit für die zeitpunktbezogene Wiederherstellung an.

a. Wiederherstellungsdatum — Wählen Sie ein Datum, das zwischen der frühesten Wiederherstellungszeit und der spätesten Wiederherstellungszeit liegt, oder geben Sie ein.

b. Wiederherstellungszeit — Wählen Sie die Stunde, Minute und Sekunden, die zwischen der frühesten Wiederherstellungszeit und der spätesten Wiederherstellungszeit liegen, oder geben Sie sie ein.

5. Füllen Sie den Bereich Configuration (Konfiguration) aus.

a. Cluster-ID — Übernehmen Sie die Standard-ID, oder geben Sie eine von Ihnen bevorzugte ID ein.

Einschränkungen bei der Benennung von Clustern:

- Die Länge besteht aus [1—63] Buchstaben, Zahlen oder Bindestrichen.
- Muss mit einem Buchstaben beginnen.
- Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.
- Muss für alle Cluster in Amazon RDS, Neptune und Amazon DocumentDB pro Region eindeutig sein. AWS-Konto

b. Instance-Klasse — Wählen Sie aus der Dropdownliste die Instance-Klasse aus, die Sie für die Instances des Clusters verwenden möchten.

c. Anzahl der Instanzen — Wählen Sie aus der Dropdownliste die Anzahl der Instanzen aus, die bei der Wiederherstellung des Clusters erstellt werden sollen.

6. Wählen Sie für die Cluster-Speicherkonfiguration eine Speicheroption.

> **Note**
>
> Die Amazon I/O-Optimized DocumentDB-Speicherkonfiguration ist nur in den Amazon DocumentDB-Engine-Versionen 5.0 und 8.0 verfügbar.

7. Optional. Um die Netzwerkeinstellungen, Cluster-Optionen und die Aktivierung von Protokollexporten zu konfigurieren, wählen Sie Show advanced settings (Erweiterte Einstellungen anzeigen) aus. Schließen Sie dann die folgenden Abschnitte ab. Fahren Sie andernfalls mit dem nächsten Schritt fort.

- Network settings (Netzwerkeinstellungen)

1. Virtual Private Cloud (VPC) — Wählen Sie aus der Dropdownliste die VPC aus, die Sie für diesen Cluster verwenden möchten.
2. Subnetzgruppe — Wählen Sie aus der Dropdownliste die Subnetzgruppe für diesen Cluster aus.
3. VPC-Sicherheitsgruppen — Wählen Sie aus der Dropdownliste die VPC-Sicherheitsgruppen für diesen Cluster aus.

- Cluster options (Cluster-Optionen)
  1. Port — Akzeptieren Sie den Standardport (27017) oder verwenden Sie die Aufwärts- und Abwärtspfeile, um den Port für die Kommunikation mit diesem Cluster festzulegen.

- Protokollexporte
  1. Audit-Logs — Wählen Sie diese Option, um den Export von Audit-Logs nach Amazon CloudWatch Logs zu ermöglichen. Wenn Sie diese Option auswählen, müssen Sie `audit_logs` in der benutzerdefinierten Parametergruppe des Clusters aktivieren. Weitere Informationen finden Sie unter Prüfung von Amazon DocumentDB-Ereignissen.
  2. Profiler-Protokolle — Wählen Sie diese Option, um den Export von Betriebsprofiler-Protokollen nach Amazon CloudWatch Logs zu aktivieren. Wenn Sie diese Option auswählen, müssen Sie auch die folgenden Parameter in der benutzerdefinierten Parametergruppe des Clusters ändern:
     - `profiler`— Auf eingestellt. `enabled`
     - `profiler_threshold_ms`— Auf einen Wert setzen`[0-INT_MAX]`, um den Schwellenwert für Profilerstellungsvorgänge festzulegen.
     - `profiler_sampling_rate`— Auf einen Wert `[0.0-1.0]` setzen, um den Prozentsatz der langsamen Operationen festzulegen, für die ein Profil erstellt werden soll.

     Weitere Informationen finden Sie unter Profilierung von Amazon DocumentDB-Vorgängen.
  3. Profiler-Protokolle — Exportieren Sie Profiler-Protokolle nach Amazon CloudWatch
  4. IAM-Rolle — Wählen Sie in der Dropdownliste die Option RDS Service Linked Role aus.

- Tags

1. Tag hinzufügen — Geben Sie im Schlüsselfeld den Namen für das Tag für Ihren Cluster ein. Geben Sie optional im Feld Value (Wert) den Tag-Wert ein. Tags werden zusammen mit AWS Identity and Access Management (IAM-) Richtlinien verwendet, um den Zugriff auf Amazon DocumentDB-Ressourcen zu verwalten und zu steuern, welche Aktionen auf die Ressourcen angewendet werden können.

- Löschschutz

1. Löschschutz aktivieren — Schützt den Cluster vor versehentlichem Löschen. Wenn diese Option aktiviert ist, können Sie den Cluster nicht löschen.

8. Um den Cluster wiederherzustellen, wählen Sie Create cluster (Cluster erstellen). Alternativ können Sie Cancel (Abbrechen) auswählen, um den Vorgang abzubrechen.

Using the AWS CLI

Um einen Cluster zu einem bestimmten Zeitpunkt mit Hilfe der Aufbewahrungsdauer des Snapshots wiederherzustellen, verwenden Sie die Operation `restore-db-cluster-to-point-in-time` mit den folgenden Parametern.

- **`--db-cluster-identifier`**— Erforderlich. Der Name des neuen Clusters, der erstellt werden soll. Dieser Cluster darf vor der Operation nicht vorhanden sein. Der Parameterwert muss den folgenden Einschränkungen entsprechen.

  Einschränkungen bei der Benennung von Clustern:

  - Die Länge beträgt [1—63] Buchstaben, Zahlen oder Bindestriche.
  - Muss mit einem Buchstaben beginnen.
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.
  - Muss für alle Cluster in Amazon RDS, Neptune und Amazon DocumentDB pro Region eindeutig sein. AWS-Konto

- **`--restore-to-time`**— Das UTC-Datum und die Uhrzeit, auf die der Cluster wiederhergestellt werden soll. Beispiel, `2018-06-07T23:45:00Z`.

  Zeiteinschränkungen:

  - Muss vor dem letzten wiederherstellbaren Zeitpunkt für den Cluster liegen.
  - Muss angegeben werden, wenn der Parameter `--use-latest-restorable-time` nicht angegeben ist.

  - Kann nicht angegeben werden, wenn der Parameter `--use-latest-restorable-time` auf `true` festgelegt ist.
  - Kann nicht angegeben werden, wenn der Parameterwert `--restore-type` `copy-on-write` ist.
- **`--source-db-cluster-identifier`**— Der Name des Quell-Clusters, aus dem wiederhergestellt werden soll. Dieser Cluster muss vorhanden und verfügbar sein.
- **`--use-latest-restorable-time`**oder **`--no-use-latest-restorable-time`** — Ob die Wiederherstellung zum letzten wiederherstellbaren Backup-Zeitpunkt erfolgen soll. Darf nicht angegeben werden, wenn der Parameter `--restore-to-time` angegeben ist.
- **`--storage-type standard | iopt1`**— Fakultativ. Standard: `standard`.

Der AWS CLI Vorgang stellt `restore-db-cluster-to-point-in-time` nur den Cluster wieder her, nicht die Instanzen für diesen Cluster. Sie müssen die Operation `create-db-instance` aufrufen, um Instances für den wiederhergestellten Cluster zu erstellen, wobei Sie in `--db-cluster-identifier` die ID des wiederhergestellten Clusters angeben. Sie können Instances erst erstellen, nachdem die Operation `restore-db-cluster-to-point-in-time` abgeschlossen wurde und wenn der wiederhergestellte Cluster verfügbar ist.

Example

Das folgende Beispiel erstellt `sample-cluster-restored` aus Snapshot `sample-cluster-snapshot` für den letzten wiederherstellbaren Zeitpunkt.

Für Linux, macOS oder Unix:

```
aws docdb restore-db-cluster-to-point-in-time \
    --db-cluster-identifier sample-cluster-restored \
    --source-db-cluster-identifier sample-cluster-snapshot \
    --use-latest-restorable-time
```

Für Windows:

```
aws docdb restore-db-cluster-to-point-in-time ^
    --db-cluster-identifier sample-cluster-restored ^
    --source-db-cluster-identifier sample-cluster-snapshot ^
    --use-latest-restorable-time
```

Example

Das folgende Beispiel erstellt `sample-cluster-restored` aus Snapshot `sample-cluster-snapshot` für 03:15 Uhr am 11. Dezember 2018 (UTC) (liegt innerhalb des Aufbewahrungszeitraums für Backups von `sample-cluster`).

Für Linux, macOS oder Unix:

```
aws docdb restore-db-cluster-to-point-in-time \
    --db-cluster-identifier sample-cluster-restore \
    --source-db-cluster-identifier sample-cluster \
    --restore-to-time 2020-05-12T03:15:00Z
```

Für Windows:

```
aws docdb restore-db-cluster-to-point-in-time ^
    --db-cluster-identifier sample-cluster-restore ^
    --source-db-cluster-identifier sample-cluster ^
    --restore-to-time 2020-05-12T03:15:00Z
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
    "DBCluster": {
        "AvailabilityZones": [
            "us-east-1c",
            "us-west-2b",
            "us-west-2a"
        ],
        "BackupRetentionPeriod": 1,
        "DBClusterIdentifier": "sample-cluster-restored",
        "DBClusterParameterGroup": "sample-parameter-group",
        "DBSubnetGroup": "default",
        "Status": "creating",
        "Endpoint": "sample-cluster-restored.node.us-east-1.docdb.amazonaws.com",
        "ReaderEndpoint": "sample-cluster-restored.node.us-
east-1.docdb.amazonaws.com",
        "MultiAZ": false,
        "Engine": "docdb",
        "EngineVersion": "4.0.0",
        "Port": 27017,
```

```
        "MasterUsername": "master-user",
        "PreferredBackupWindow": "02:00-02:30",
        "PreferredMaintenanceWindow": "tue:09:50-tue:10:20",
        "DBClusterMembers": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abc0123",
                "Status": "active"
            }
        ],
        "HostedZoneId": "ABCDEFGHIJKLM",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID^>:key/sample-key",
        "DbClusterResourceId": "cluster-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
        "DBClusterArn": "arn:aws:rds:us-east-1:<accountID>:cluster:sample-cluster-
restored",
        "AssociatedRoles": [],
        "ClusterCreateTime": "2020-04-24T20:14:36.713Z",
        "DeletionProtection": false
    }
}
```

Löschen eines Cluster-Snapshots

Ein manueller Snapshot ist ein vollständiges Backup, das nur gelöscht wird, wenn Sie es manuell mit dem AWS-Managementkonsole oder AWS CLI löschen. Sie können einen automatischen Snapshot nicht manuell löschen, da automatische Snapshots nur gelöscht werden, wenn die Aufbewahrungsfrist des Snapshots abläuft oder Sie den Cluster des Snapshots löschen.

Using the AWS-Managementkonsole

Gehen Sie wie folgt vor AWS-Managementkonsole, um einen manuellen Cluster-Snapshot mithilfe von zu löschen.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Wählen Sie im Navigationsbereich die Option Snapshots.

> **Tip**
>
> Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Wählen Sie in der Liste der Snapshots die Schaltfläche links neben dem Snapshot, den Sie löschen möchten. Der Typ des Snapshots muss manuell sein.

   1. Sie können überprüfen, ob der Snapshot-Typ manuell ist, indem Sie überprüfen, ob er als `manual` oder `automatic` unter der Spalte Typ aufgeführt ist.

4. Wählen Sie im Menü Actions (Aktionen) die Option Delete (Löschen) aus. Wenn die Option Delete (Löschen) nicht verfügbar ist, haben Sie wahrscheinlich einen automatischen Snapshot auswählt.
5. Um den Snapshot zu löschen, wählen Sie auf der Bestätigungsseite des Löschvorgangs Delete (Löschen) aus. Um den Snapshot zu erhalten, wählen Sie Cancel (Abbrechen) aus.

Using the AWS CLI

Ein manueller Amazon DocumentDB-Cluster-Snapshot ist ein vollständiges Backup, das Sie mithilfe von manuell löschen können. AWS CLI Sie können einen automatischen Snapshot nicht manuell löschen.

Um einen manuellen Cluster-Snapshot mithilfe von zu löschen AWS CLI, verwenden Sie den `delete-db-cluster-snapshot` Vorgang mit den folgenden Parametern.

Parameters

- **`--db-cluster-snapshot-identifier`** – Erforderlich. Der Name des zu löschenden manuellen Snapshots.

Das folgende Beispiel löscht den Cluster-Snapshot `sample-cluster-snapshot`.

Für Linux, macOS oder Unix:

```
aws docdb delete-db-cluster-snapshot \
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

Für Windows:

```
aws docdb delete-db-cluster-snapshot ^
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

Die Ausgabe dieses Vorgangs führt die Einzelheiten des von Ihnen gelöschten Cluster-Snapshots auf.

Überwachung und Protokollierung in Amazon DocumentDB

Die Überwachung Ihrer AWS Dienste ist ein wichtiger Bestandteil, damit Ihre Systeme gesund bleiben und optimal funktionieren. Erfassen Sie Überwachungsdaten aus allen Teilen Ihrer AWS Lösung, damit Sie Fehler oder Leistungsprobleme einfacher debuggen und beheben können. Bevor Sie mit der Überwachung Ihrer AWS Lösungen beginnen, sollten Sie sich die folgenden Fragen stellen:

- Was sind Ihre Überwachungsziele?
- Welche Ressourcen werden Sie überwachen?
- Wie oft werden Sie diese Ressourcen überwachen?
- Welche Überwachungstools werden verwendet?
- Wer ist für die Überwachung verantwortlich?
- Wer ist zu benachrichtigen und auf welche Weise, wenn etwas schief geht?

Um Ihre aktuellen Leistungsmuster zu verstehen, Leistungsanomalien zu identifizieren und Methoden zur Behebung von Problemen zu formulieren, sollten Sie grundlegende Leistungskennzahlen für verschiedene Zeiten und unter unterschiedlichen Lastbedingungen festlegen. Speichern Sie bei der Überwachung Ihrer AWS Lösung Ihre historischen Überwachungsdaten, damit Sie später darauf zurückgreifen und Ihre Basiswerte festlegen können.

Die zulässigen Werte für Leistungsmetriken sind im Allgemeinen davon abhängig, wie die Ausgangsbasis aussieht und wofür die Anwendung gedacht ist. Prüfen Sie, ob dauerhafte oder tendenzielle Abweichungen von Ihrer Ausgangsbasis vorliegen. Im Folgenden werden Hinweise zu bestimmten Arten von Kennzahlen gegeben:

- Hohe CPU- oder RAM-Auslastung — Hohe Werte für die CPU- oder RAM-Auslastung könnten angemessen sein, vorausgesetzt, sie entsprechen Ihren Zielen für Ihre Anwendung (wie Durchsatz oder Parallelität) und werden erwartet.
- Speichervolumenverbrauch — Untersuchen Sie den Speicherverbrauch (`VolumeBytesUsed`), wenn der verwendete Speicherplatz konstant mindestens 85 Prozent des gesamten Speichervolumens ausmacht. Legen Sie fest, ob Sie Daten aus dem Speicher-Volume löschen oder Daten in ein anderes System archivieren können, um Platz zu schaffen. Weitere Informationen erhalten Sie unter Amazon DocumentDB-Speicher und Amazon DocumentDB-Kontingente.

- Netzwerkverkehr — Sprechen Sie mit Ihrem Systemadministrator, um zu erfahren, wie hoch der erwartete Durchsatz für Ihr Domänennetzwerk und Ihre Internetverbindung ist. Überprüfen Sie den Netzwerkdatenverkehr, wenn der Durchsatz dauerhaft unter dem erwarteten Wert liegt.
- Datenbankverbindungen — Erwägen Sie, Datenbankverbindungen einzuschränken, wenn Sie eine hohe Anzahl von Benutzerverbindungen in Verbindung mit einer Verringerung der Instance-Leistung und Reaktionszeit feststellen. Die optimale Anzahl der Benutzerverbindungen für Ihre Instance ist von der Instance-Klasse und der Komplexität der Operationen abhängig, die ausgeführt werden.
- IOPS-Metriken — Die erwarteten Werte für IOPS-Metriken hängen von der Festplattenspezifikation und der Serverkonfiguration ab. Verwenden Sie daher Ihren Basiswert, um zu ermitteln, was typisch ist. Prüfen Sie, ob dauerhafte Abweichungen von den Werten Ihrer Ausgangsbasis vorliegen. Für eine optimale IOPS-Leistung stellen Sie sicher, dass Ihr typisches Working Set in den Speicher passt, um Lese- und Schreibvorgänge zu minimieren.

Amazon DocumentDB (mit MongoDB-Kompatibilität) bietet eine Vielzahl von CloudWatch Amazon-Metriken, die Sie überwachen können, um den Zustand und die Leistung Ihrer Amazon DocumentDB-Cluster und -Instances zu ermitteln. Sie können Amazon DocumentDB-Metriken mithilfe verschiedener Tools anzeigen, darunter die Amazon DocumentDB-Konsole AWS CLI, CloudWatch API und Performance Insights.

Themen

- Überwachung des Status eines Amazon DocumentDB-Clusters
- Überwachung des Status einer Amazon DocumentDB-Instance
- Empfehlungen von Amazon DocumentDB anzeigen
- Amazon DocumentDB DocumentDB-Veranstaltungsabonnements verwenden
- Überwachung von Amazon DocumentDB mit CloudWatch
- Protokollieren Amazon DocumentDB DocumentDB-API-Aufrufen mit AWS CloudTrail
- Profilierung von Amazon DocumentDB-Vorgängen
- Überwachung mit Performance Insights

Überwachung des Status eines Amazon DocumentDB-Clusters

Der Status eines Clusters zeigt den Zustand des Clusters an. Sie können den Status eines Clusters mithilfe der Amazon DocumentDB-Konsole oder des AWS CLI `describe-db-clusters` Befehls anzeigen.

Themen

- Werte für den Cluster-Status
- Überwachung des Status eines Clusters

Werte für den Cluster-Status

In der folgenden Tabelle sind die gültigen Werte für den Cluster-Status aufgeführt.

Cluster-Status	Description
`active`	Der Cluster ist aktiv. Dieser Status gilt nur für elastische Cluster.
`available`	Der Cluster ist stabil und verfügbar. Dieser Status gilt nur für instanzbasierte Cluster.
`backing-up`	Der Cluster wird derzeit gesichert.
`creating`	Der Cluster wird gerade erstellt. Ist während der Erstellung unzugänglich.
`deleting`	Der Cluster wird gerade gelöscht. Ist während des Löschvorgangs nicht zugreifbar.
`failing-over`	Ein Failover von der primären Instance zu einer Amazon

Cluster-Status	Description
	DocumentDB-Replica wird durchgeführt.
`inaccessible-encryption-credentials`	Auf den AWS KMS Schlüssel , der zum Verschlüsseln oder Entschlüsseln des Clusters verwendet wurde, kann nicht zugegriffen werden.
`maintenance`	Ein Wartungsupdate wird auf den Cluster angewende t. Dieser Status wird für Wartungsarbeiten auf Cluster-Ebene verwendet, die Amazon DocumentDB weit im Voraus plant.
`migrating`	Es wird gerade ein Cluster-Snapshot in einen Cluster wiederhergestellt.
`migration-failed`	Eine Migration ist fehlgesch lagen.
`modifying`	Der Cluster wird aufgrund einer Kundenanfrage, den Cluster zu ändern, geändert.
`renaming`	Der Cluster wird aufgrund einer Kundenanfrage auf Instance-Umbenennung umbenannt.
`resetting-master-credentials`	Die Master-Anmeldeinfo rmationen für den Cluster werden auf eine Kundenanf rage auf Zurücksetzung hin zurückgesetzt.

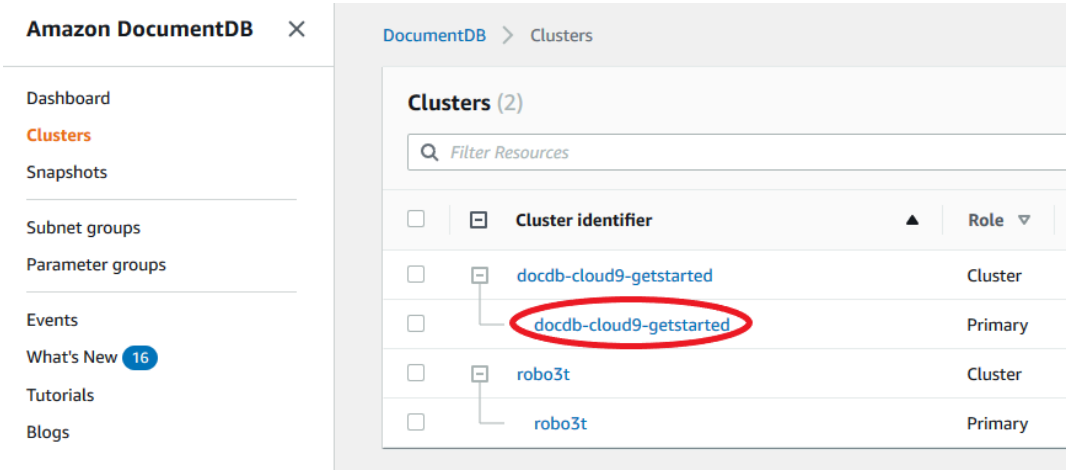
Cluster-Status	Description
`upgrading`	Die Cluster-Engine wird auf eine neue Version aktualisiert.

Überwachung des Status eines Clusters

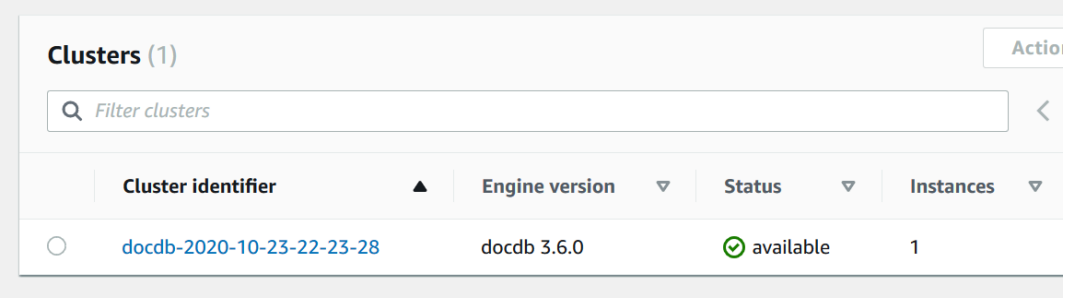
Using the AWS-Managementkonsole

Wenn Sie den verwenden AWS-Managementkonsole , um den Status eines Clusters zu ermitteln, gehen Sie wie folgt vor.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Klicken Sie im Navigationsbereich auf Cluster.
3. Im Navigationsfeld „Cluster“ wird die Spalte Cluster-ID angezeigt. Ihre Instances sind unter Clustern aufgeführt, ähnlich wie im folgenden Screenshot.



4. Suchen Sie in der Spalte Cluster-ID nach dem Namen der Instanz, an der Sie interessiert sind. Um den Status der Instanz zu ermitteln, lesen Sie dann von dieser Zeile zur Statusspalte über, wie unten dargestellt.



Using the AWS CLI

Wenn Sie den verwenden AWS CLI , um den Status eines Clusters zu ermitteln, verwenden Sie den `describe-db-clusters` Vorgang. Der folgende Code ermittelt den Status des Clusters `sample-cluster`.

Für Linux, macOS oder Unix:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster  \
    --query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

Für Windows:

```
aws docdb describe-db-clusters ^
    --db-cluster-identifier sample-cluster  ^
    --query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
[
    [
        "sample-cluster",
        "available"
    ]
]
```

Überwachung des Status einer Amazon DocumentDB-Instance

Amazon DocumentDB bietet Informationen über den aktuellen Zustand jeder konfigurierten Instanz in der Datenbank.

Es gibt drei Arten von Status, die Sie für eine Amazon DocumentDB-Instance einsehen können:

- Instanzstatus: Dieser Status wird in der Statusspalte der Cluster-Tabelle in der angezeigt AWS-Managementkonsole und zeigt den aktuellen Lebenszyklusstatus der Instance. Die in der Status-Spalte angezeigten Werte werden aus dem `Status` Feld der `DescribeDBCluster` API-Antwort abgeleitet.
- Integritätsstatus der Instanz: Dieser Status wird in der Spalte Instanzstatus der Cluster-Tabelle in der angezeigt AWS-Managementkonsole und zeigt, ob die Datenbank-Engine, die Komponente, die für die Verwaltung und das Abrufen von Daten verantwortlich ist, läuft. Die in der Spalte Instance-Zustand angezeigten Werte basieren auf der CloudWatch `EngineUptime` Amazon-Systemmetrik.
- Wartungsstatus: Dieser Status wird in der Spalte Wartung der Cluster-Tabelle in der angezeigt AWS-Managementkonsole und gibt den Status aller Wartungsereignisse an, die auf eine Instance angewendet werden müssen. Der Wartungsstatus ist unabhängig vom Status der anderen Instanzen und wird von der `PendingMaintenanceAction` API abgeleitet. Weitere Informationen zum Wartungsstatus finden Sie unter Wartung von Amazon DocumentDB.

Themen

- Instance-Statuswerte
- Überwachen Sie den Instanzstatus mithilfe der AWS-Managementkonsole oder AWS CLI
- Werte für den Integritätsstatus der Instanz
- Überwachen Sie den Integritätsstatus der Instanz mit dem AWS-Managementkonsole

Instance-Statuswerte

Die folgende Tabelle führt die möglichen Statuswerte für Instances auf und gibt an, wie welcher Status berechnet wird. Die Tabelle gibt an, ob Sie für Instance und Speicher, nur für Speicher oder weder für Instance noch Speicher zahlen müssen. Für alle Instance-Status wird immer die Sicherungsnutzung berechnet.

Instance-Status	Berechnet	Description
`available`	Berechnet	Die Instance ist stabil und verfügbar.
`backing-up`	Berechnet	Die Instance wird derzeit gesichert.

Instance-Status	Berechnet	Description
`configuring-log-exports`	Berechnet	Die Veröffentlichung von Protokolldateien in Amazon CloudWatch Logs wird für diese Instance aktiviert oder deaktiviert.
`creating`	Nicht berechnet	Die Instance wird gerade erstellt. Während die Instance erstellt wird, kann nicht auf sie zugegriffen werden.
`deleting`	Nicht berechnet	Die Instance wird gerade gelöscht.
`failed`	Nicht berechnet	Die Instance ist ausgefallen und Amazon DocumentDB konnte sie nicht wiederher stellen. Um die Daten wiederherzustellen, führen Sie eine Point-in-Time-Wiederherstel lung auf den letzten wiederherstellbaren Zeitpunkt der Instance durch.
`inaccessible-encryption-cre dentials`	Nicht berechnet	Auf den AWS KMS Schlüssel, der zum Verschlüsseln oder Entschlüsseln der Instanz verwendet wird, konnte nicht zugegriffen werden.
`incompatible-network`	Nicht berechnet	Amazon DocumentDB versucht, eine Wiederherstellungsaktion für eine Instance durchzuführen, kann dies jedoch nicht tun, da sich die VPC in einem Zustand befindet, der den Abschluss der Aktion verhindert. Dieser Status kann auftreten, wenn beispiels weise alle verfügbaren IP-Adressen in einem Subnetz verwendet wurden und Amazon DocumentDB keine IP-Adresse für die Instance abrufen konnte.

Instance-Status	Berechnet	Description
`maintenance`	Berechnet	Amazon DocumentDB wendet ein Wartungsupdate auf die Instance an. Dieser Status wird für Wartungsarbeiten auf Instance-Ebene verwendet, die Amazon DocumentDB weit im Voraus plant. Wir evaluieren derzeit Möglichkeiten, Kunden über diesen Status auch weitere Wartungsaktionen kenntlich zu machen.
`modifying`	Berechnet	Die Instance wird aufgrund einer Anforderung zur Änderung der Instance geändert.
`rebooting`	Berechnet	Die Instance wird aufgrund einer Anfrage oder eines Amazon DocumentDB-Prozesses neu gestartet, der einen Neustart der Instance erfordert.
`renaming`	Berechnet	Die Instance wird aufgrund einer Anforderung zur Umbenennung umbenannt.
`resetting-master-credentials`	Berechnet	Die Master-Anmeldeinformationen für die Instance werden aufgrund einer Anforderung zum Zurücksetzen zurückgesetzt.
`restore-error`	Berechnet	Bei dem Versuch, die Instance anhand eines Wiederherstellungszeitpunkts oder eines Snapshots wiederherzustellen, ist ein Fehler aufgetreten.
`starting`	Berechnet für Speicher	Die Instance wird gestartet.
`stopped`	Berechnet für Speicher	Die Instance wird angehalten.
`stopping`	Berechnet für Speicher	Die Instance wird gestoppt.

Instance-Status	Berechnet	Description
`storage-full`	Berechnet	Die Instance hat seine Speicherkapazitäts zuteilung erreicht. Dieser Status ist kritisch und sollte sofort behoben werden. Sie müssen den verfügbaren Speicherplatz hochskalieren, indem Sie die Instance entsprechend ändern. Stellen Sie CloudWatch Amazon-Alarme ein, um Sie zu warnen, wenn der Speicherplatz knapp wird, damit Sie nicht in diese Situation geraten.

Überwachen Sie den Instanzstatus mithilfe der AWS-Managementkonsole oder AWS CLI

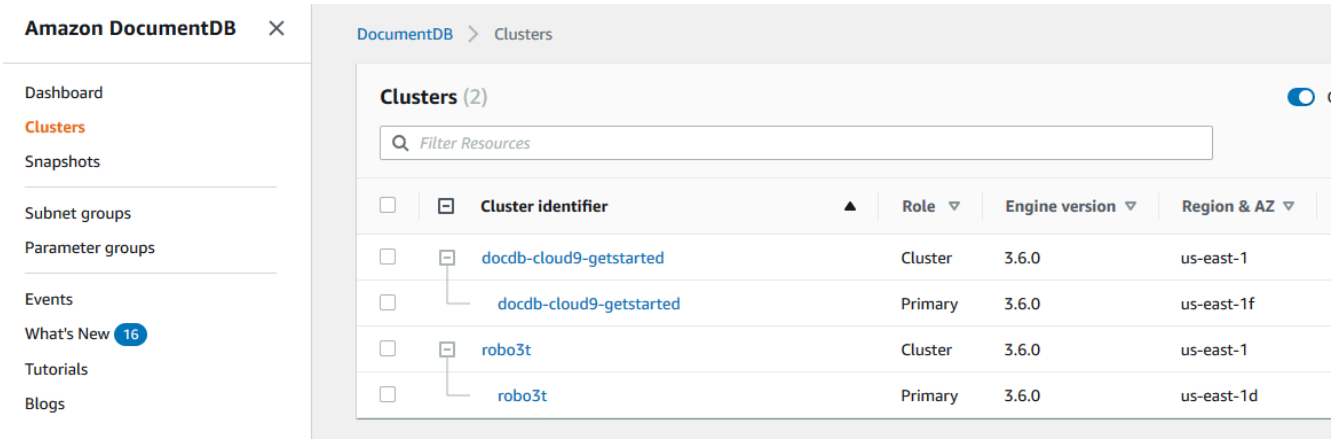
Verwenden Sie AWS-Managementkonsole oder AWS CLI , um den Status Ihrer Instanz zu überwachen.

Using the AWS-Managementkonsole

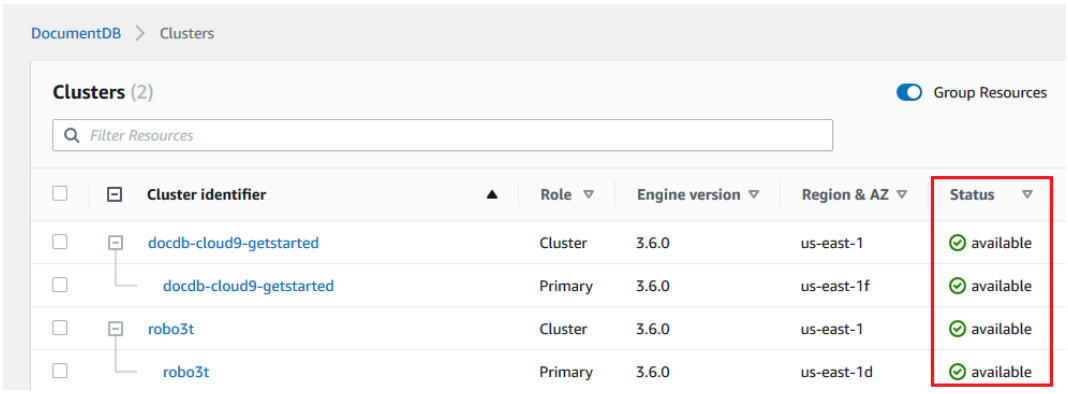
Wenn Sie den verwenden AWS-Managementkonsole , um den Status eines Clusters zu ermitteln, gehen Sie wie folgt vor.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Klicken Sie im Navigationsbereich auf Cluster.

> **Note**
>
> Beachten Sie, dass im Navigationsfeld Cluster in der Spalte Cluster-Identifier sowohl Cluster als auch Instances angezeigt werden. Instanzen werden unter Clustern aufgeführt, ähnlich wie in der folgenden Abbildung.



3. Suchen Sie den Namen der Instance, an der Sie interessiert sind. Um den Status der Instance zu ermitteln, prüfen Sie die Zeile in die Spalte Status.



Using the AWS CLI

Wenn Sie den verwenden AWS CLI , um den Status eines Clusters zu ermitteln, verwenden Sie den `describe-db-instances` Vorgang. Der folgende Code ermittelt den Status der Instance `sample-cluster-instance-01`.

Für Linux, macOS oder Unix:

```
aws docdb describe-db-instances \
          --db-instance-identifier sample-cluster-instance-01  \
          --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceStatus]'
```

Für Windows:

```
aws docdb describe-db-instances ^
```

```
          --db-instance-identifier sample-cluster-instance-01  ^
          --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceStatus]'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
[
          [
              "sample-cluster-instance-01",
              "available"
          ]
      ]
```

Werte für den Integritätsstatus der Instanz

In der folgenden Tabelle sind die möglichen Integritätsstatuswerte für Instances aufgeführt. Die Spalte Instanzstatus, die sich in der Cluster-Tabelle unter befindet AWS-Managementkonsole, zeigt, ob die Datenbank-Engine, die Komponente, die für das Speichern, Verwalten und Abrufen von Daten verantwortlich ist, normal funktioniert. In dieser Spalte wird auch angegeben, ob die `EngineUptime` Systemmetrik, die in verfügbar ist CloudWatch, den Integritätsstatus jeder Instance anzeigt.

Instance-Zustand	Description
fehlerfrei	Die Datenbank-Engine läuft in der Amazon DocumentDB-Instance.
ungesund	Das Datenbankmodul läuft nicht oder wurde vor weniger als einer Minute neu gestartet.

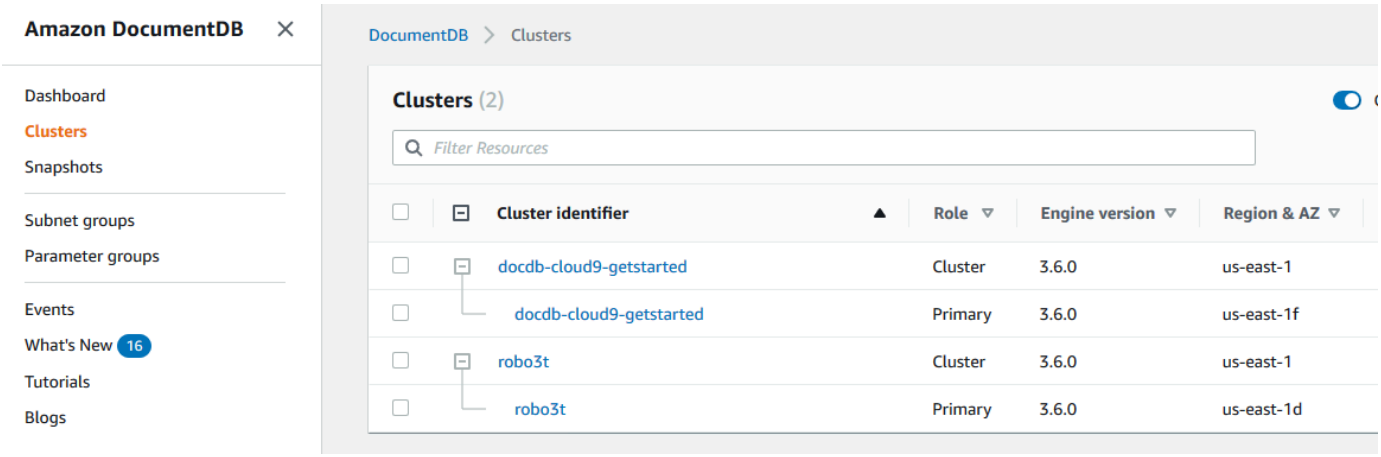
Überwachen Sie den Integritätsstatus der Instanz mit dem AWS-Managementkonsole

Verwenden Sie den AWS-Managementkonsole , um den Integritätsstatus Ihrer Instance zu überwachen.

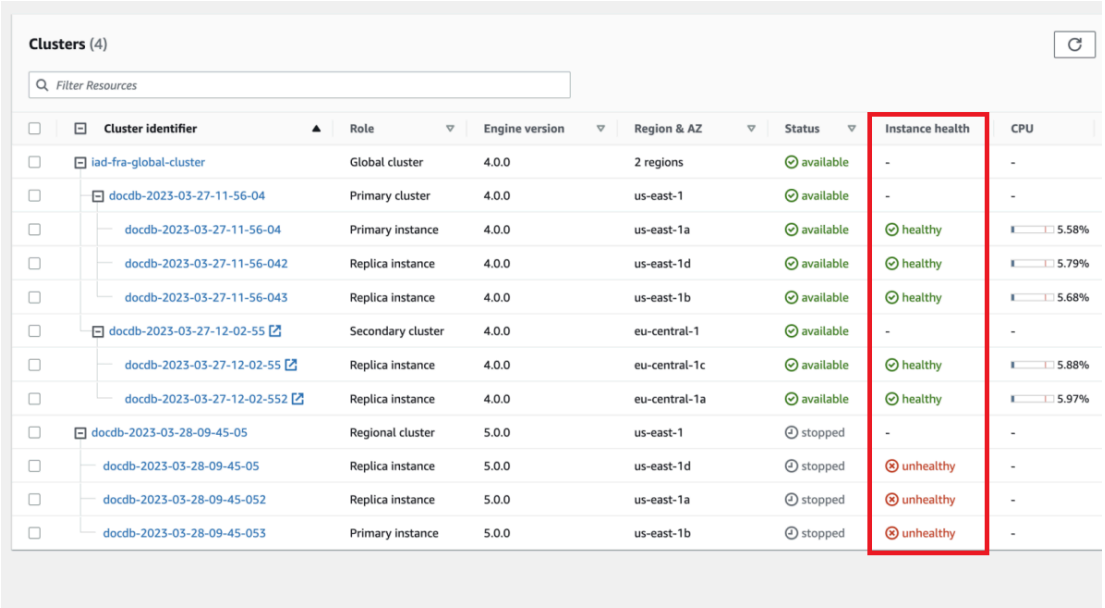
Verwenden Sie bei der Verwendung die folgenden Schritte AWS-Managementkonsole, um den Integritätsstatus der Instanz zu verstehen.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Klicken Sie im Navigationsbereich auf Cluster.

> **Note**
>
> Im Navigationsfeld Cluster werden in der Spalte Cluster-Identifier sowohl Cluster als auch Instances angezeigt. Instanzen werden unter Clustern aufgeführt, ähnlich wie in der folgenden Abbildung.



3. Suchen Sie den Namen der Instance, an der Sie interessiert sind. Um dann den Status der Instance zu ermitteln, lesen Sie in dieser Zeile die Spalte Instance-Status durch, wie in der folgenden Abbildung dargestellt:



> **Note**
>
> Die Abfrage des Instanzzustands erfolgt alle 60 Sekunden und basiert auf der CloudWatch `EngineUptime` Systemmetrik. Die Werte in der Spalte Instanzstatus werden automatisch aktualisiert.

Empfehlungen von Amazon DocumentDB anzeigen

Amazon DocumentDB bietet eine Liste automatisierter Empfehlungen für Datenbankressourcen wie Instances und Cluster. Diese Empfehlungen bieten bewährte Anleitungen zur Analyse Ihrer Cluster- und Instance-Konfigurationen.

Ein Beispiel für diese Empfehlungen finden Sie im Folgenden:

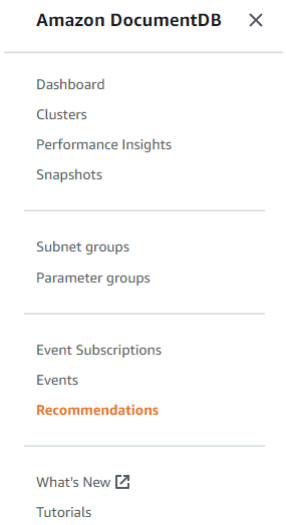
Typ	Beschreibung	Empfehlung	Zusätzliche Informationen
Eine -Instance	Cluster enthält nur eine Instanz	Leistung und Verfügbarkeit: Fügen Sie eine weitere Instanz mit derselben Instanzklasse in einer	Hohe Verfügbarkeit und Replikation von Amazon DocumentDB

Typ	Beschreibung	Empfehlung	Zusätzliche Informationen
		anderen Availability Zone hinzu.	

Amazon DocumentDB generiert Empfehlungen für eine Ressource, wenn die Ressource erstellt oder geändert wird. Amazon DocumentDB scannt außerdem regelmäßig Ihre Ressourcen und generiert Empfehlungen.

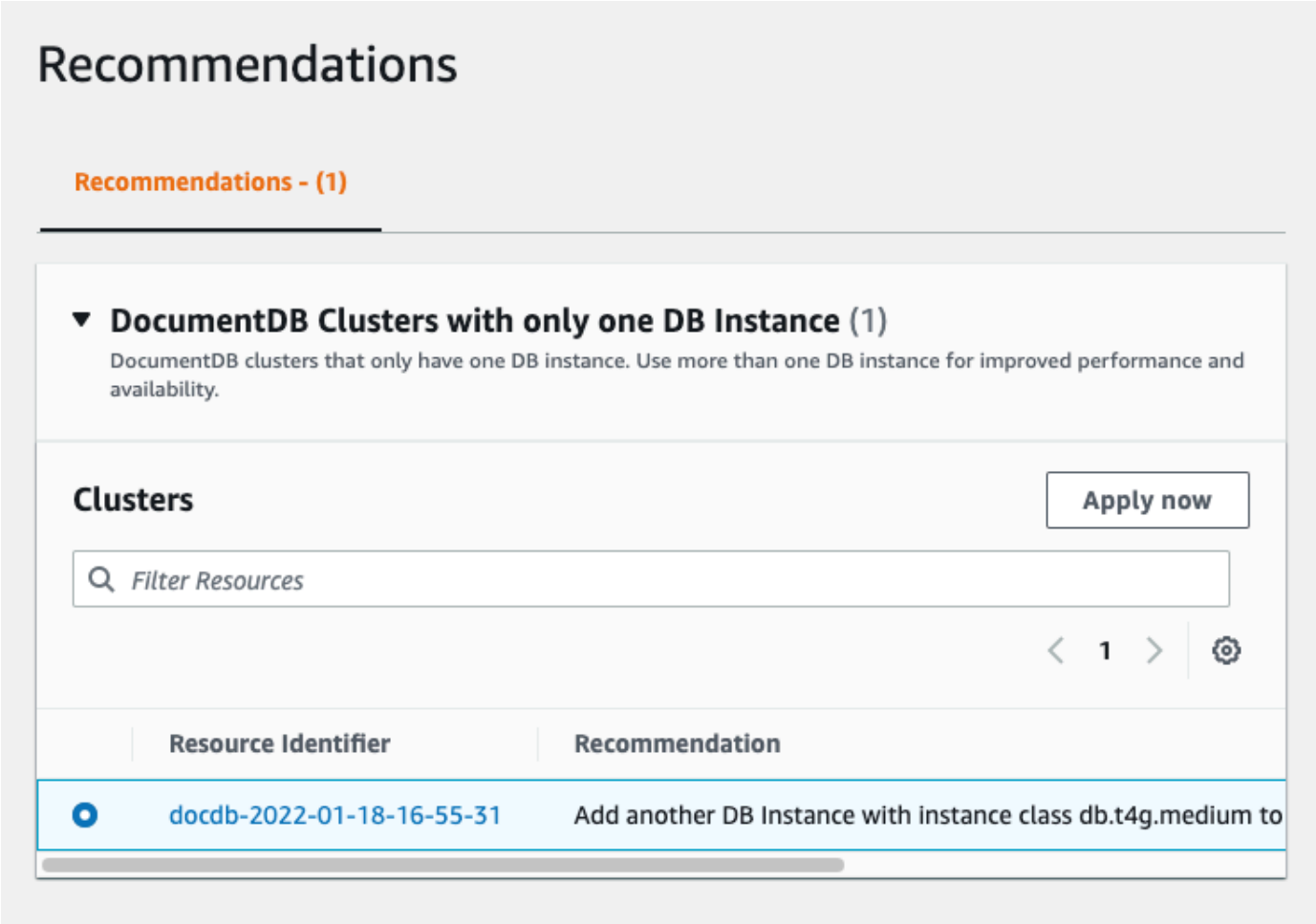
Um die Empfehlungen von Amazon DocumentDB einzusehen und entsprechende Maßnahmen zu ergreifen

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Wählen Sie im Navigationsbereich Empfehlungen aus:



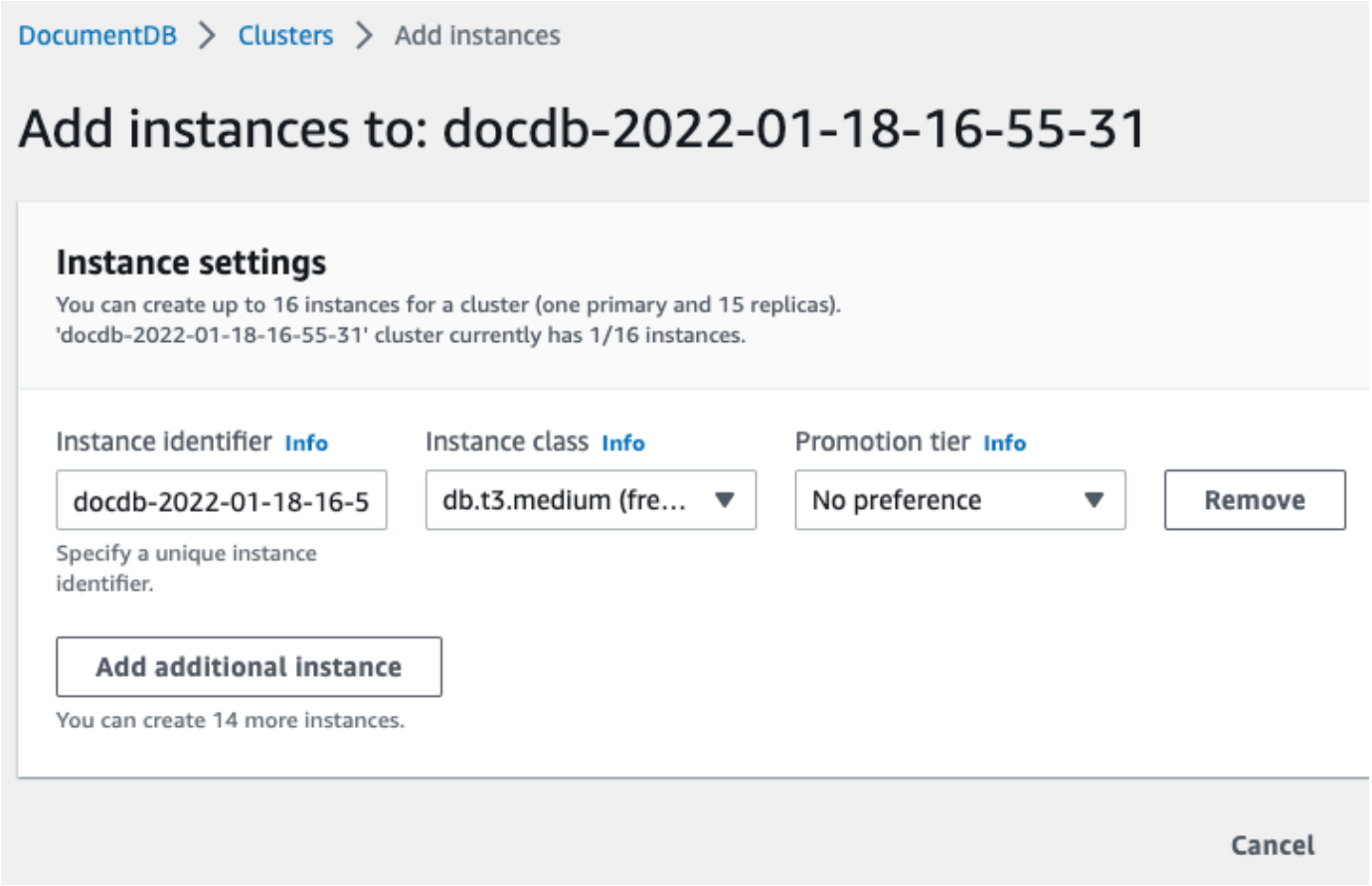
3. Erweitern Sie im Dialogfeld „Empfehlungen“ den gewünschten Bereich und wählen Sie die empfohlene Aufgabe aus.

   Im folgenden Beispiel gilt die empfohlene Aufgabe für einen Amazon DocumentDB-Cluster mit nur einer Instance. Es wird empfohlen, eine weitere Instance hinzuzufügen, um die Leistung und Verfügbarkeit zu verbessern.



4. Wählen Sie Apply now (Jetzt anwenden) aus.

   In diesem Beispiel wird das Dialogfeld „Instanzen hinzufügen“ angezeigt:



5. Ändern Sie die Einstellungen Ihrer neuen Instanz und wählen Sie Erstellen.

Amazon DocumentDB DocumentDB-Veranstaltungsabonnements verwenden

Amazon DocumentDB verwendet Amazon Simple Notification Service (Amazon SNS), um Benachrichtigungen bereitzustellen, wenn ein Amazon DocumentDB DocumentDB-Ereignis eintritt. Diese Benachrichtigungen können in jeder Form erfolgen, die von Amazon SNS unterstützt wird AWS-Region, z. B. als E-Mail, Textnachricht oder Anruf an einen HTTP-Endpunkt.

Amazon DocumentDB gruppiert diese Ereignisse in Kategorien, die Sie abonnieren können, sodass Sie benachrichtigt werden können, wenn ein Ereignis in dieser Kategorie eintritt. Sie können eine Ereigniskategorie für eine Instance, einen Cluster, einen Snapshot, einen Cluster-Snapshot oder eine Parametergruppe abonnieren. Wenn Sie beispielsweise die Backup-Kategorie für eine bestimmte Instanz abonnieren, werden Sie jedes Mal benachrichtigt, wenn ein Backup-bezogenes Ereignis eintritt, das sich auf die Instanz auswirkt. Sie erhalten auch eine Benachrichtigung, wenn sich ein Abonnement für ein Ereignis ändert.

Ereignisse treten sowohl auf Cluster- als auch auf Instanzebene auf, sodass Sie Ereignisse empfangen können, wenn Sie einen Cluster oder eine Instance abonnieren.

Ereignisabonnements werden an die Adressen gesendet, die Sie bei der Erstellung des Abonnements angegeben haben. Möglicherweise möchten Sie mehrere verschiedene Abonnements erstellen, z. B. ein Abonnement, das alle Ereignisbenachrichtigungen empfängt, und ein anderes Abonnement, das nur kritische Ereignisse für Ihre Produktionsinstanzen enthält. Sie können die Benachrichtigungen ganz einfach deaktivieren, ohne ein Abonnement zu löschen. Setzen Sie dazu das Optionsfeld Aktiviert in der Amazon DocumentDB DocumentDB-Konsole auf Nein.

> **Important**
>
> Amazon DocumentDB garantiert nicht die Reihenfolge der Ereignisse, die in einem Event-Stream gesendet werden. Die Reihenfolge der Ereignisse kann sich ändern.

Amazon DocumentDB verwendet den Amazon-Ressourcennamen (ARN) eines Amazon SNS-Themas, um jedes Abonnement zu identifizieren. Die Amazon DocumentDB DocumentDB-Konsole erstellt den ARN für Sie, wenn Sie das Abonnement erstellen.

Die Abrechnung von Amazon DocumentDB DocumentDB-Veranstaltungsabonnements erfolgt über Amazon SNS. Bei Verwendung von Ereignisbenachrichtigungen fallen Amazon-SNS-Gebühren an. Weitere Informationen finden Sie unter Preise für den Amazon Simple Notification Service (SNS). Abgesehen von den Amazon SNS SNS-Gebühren stellt Amazon DocumentDB keine Event-Abonnements in Rechnung.

Themen

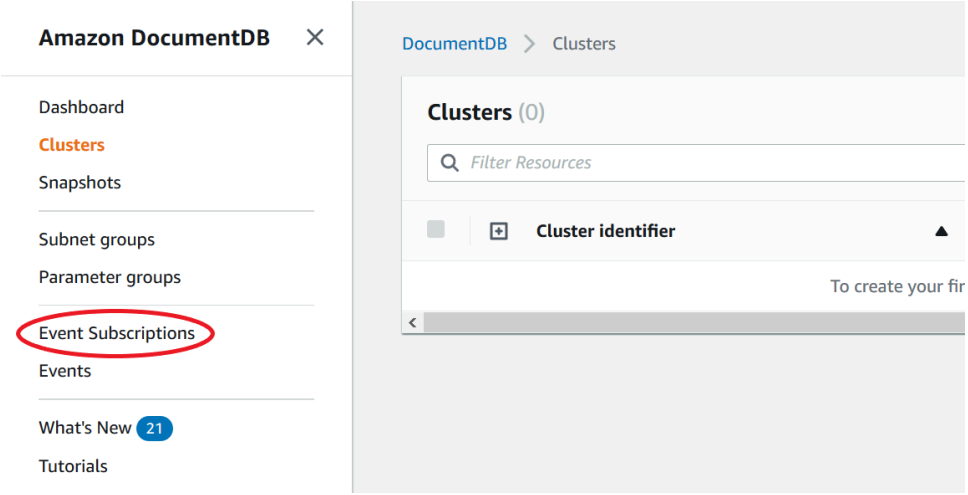
- Amazon DocumentDB DocumentDB-Veranstaltungen abonnieren
- Verwaltung von Abonnements für Amazon DocumentDB DocumentDB-Ereignisbenachrichtigungen
- Amazon DocumentDB DocumentDB-Ereigniskategorien und Meldungen

Amazon DocumentDB DocumentDB-Veranstaltungen abonnieren

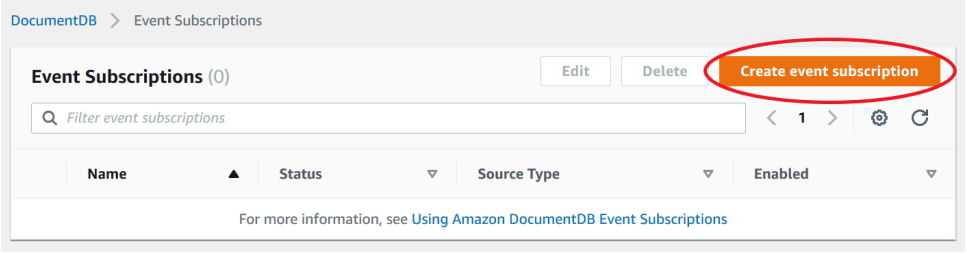
Sie können die Amazon DocumentDB DocumentDB-Konsole verwenden, um Veranstaltungsabonnements wie folgt zu abonnieren:

1. Melden Sie sich AWS-Managementkonsole unter an. https://console.aws.amazon.com/docdb

2. Wählen Sie im Navigationsbereich Ereignisabonnements aus.

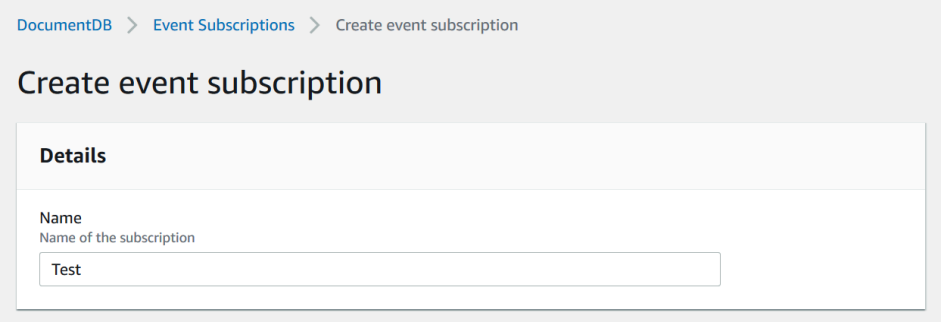


3. Wählen Sie im Bereich Ereignisabonnements Ereignisabonnement erstellen aus.

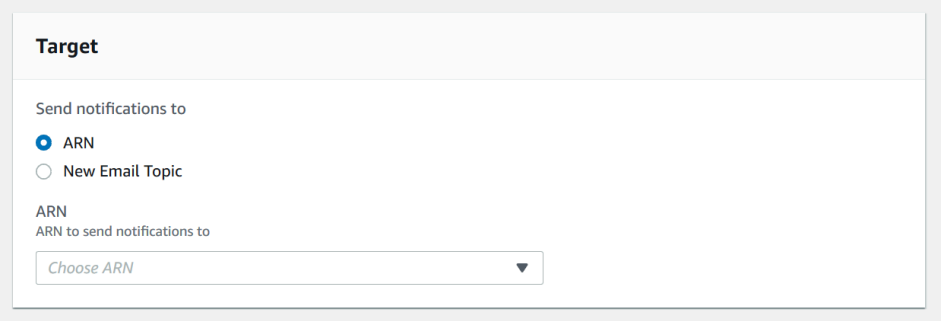


4. Gehen Sie im Dialogfeld Ereignisabonnement erstellen wie folgt vor:

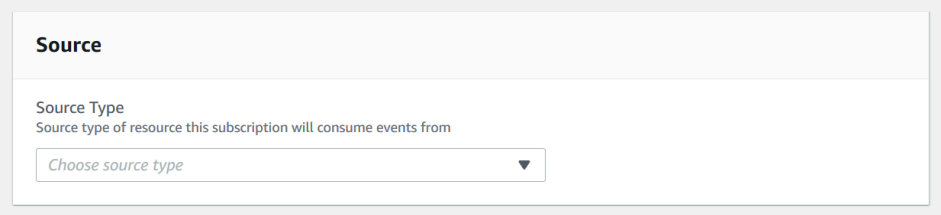
   - Geben Sie unter Name einen Namen für das Abonnement für Ereignisbenachrichtigungen ein.



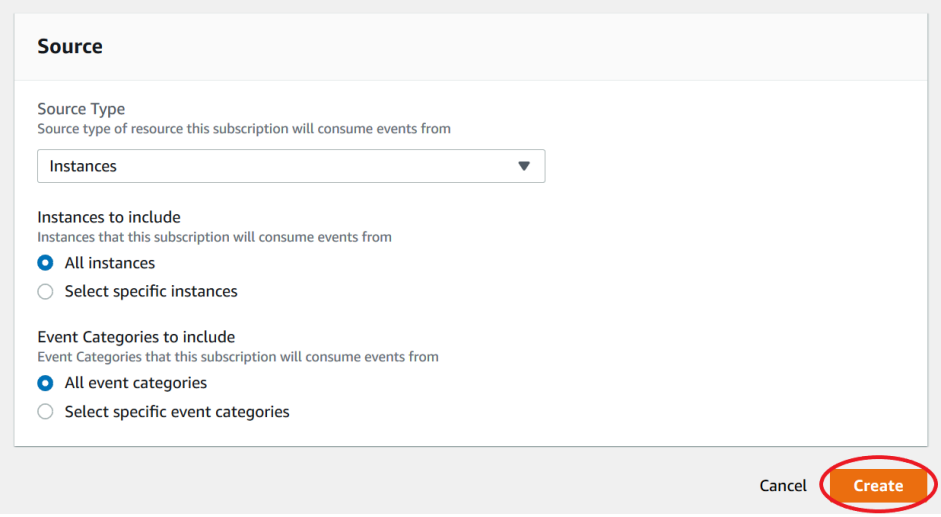
   - Wählen Sie für Target aus, wohin Sie Benachrichtigungen senden möchten. Sie können einen vorhandenen ARN oder Neues E-Mail-Thema wählen, um den Namen eines Themas und eine Empfängerliste einzugeben.



- Wählen Sie für Quelle einen Quelltyp aus. Wählen Sie abhängig vom ausgewähltem Quelltyp die Ereigniskategorien und Quellen aus, für die Sie Ereignisbenachrichtigungen erhalten möchten.



- Wählen Sie Erstellen aus.

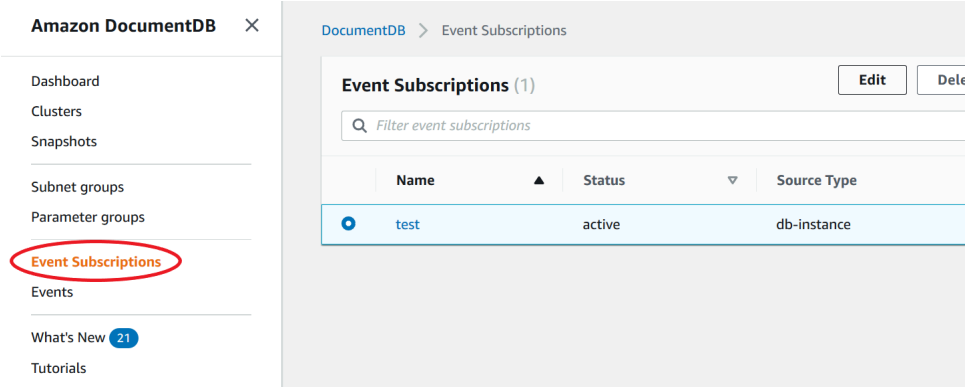


Verwaltung von Abonnements für Amazon DocumentDB DocumentDB-Ereignisbenachrichtigungen

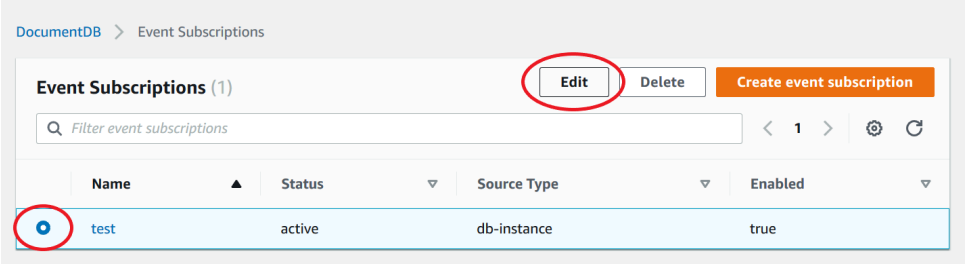
Wenn Sie im Navigationsbereich der Amazon DocumentDB DocumentDB-Konsole Event-Abonnements auswählen, können Sie Abonnementkategorien und eine Liste Ihrer aktuellen Abonnements einsehen. Sie können auch ein bestimmtes Abonnement ändern oder löschen.

So ändern Sie Ihre aktuellen Amazon DocumentDB DocumentDB-Abonnements für Ereignisbenachrichtigungen

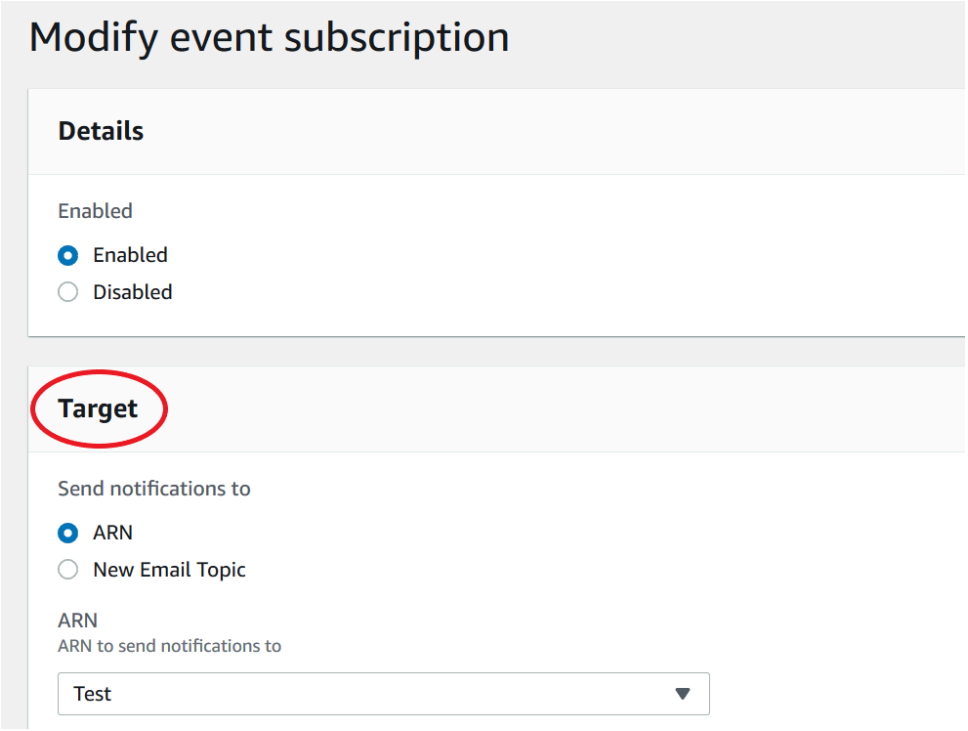
1. Melden Sie sich AWS-Managementkonsole unter an. https://console.aws.amazon.com/docdb
2. Wählen Sie im Navigationsbereich Ereignisabonnements aus. Im Bereich Ereignisabonnements werden all Ihre Abonnements für Ereignisbenachrichtigungen angezeigt.



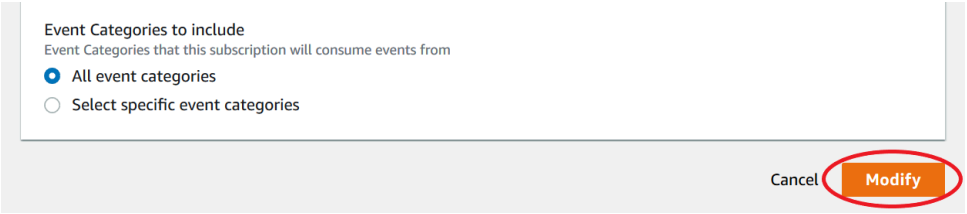
3. Wählen Sie im Bereich Ereignisabonnements das Abonnement, das Sie modifizieren möchten, und klicken Sie auf Bearbeiten.



4. Nehmen Sie Ihre Änderungen am Abonnement im Bereich Ziel oder Quelle vor. Sie können Quell-IDs hinzufügen oder entfernen, indem Sie diese im Bereich Quelle auswählen oder ihre Auswahl aufheben.



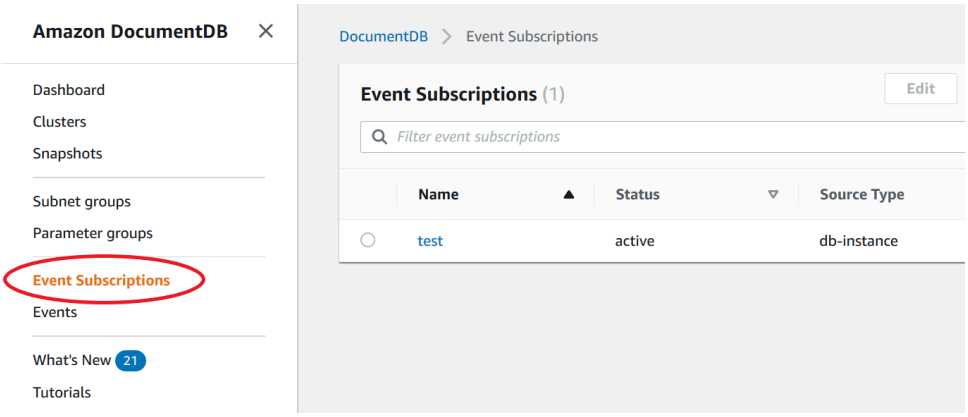
5. Wählen Sie Ändern aus. Die Amazon DocumentDB DocumentDB-Konsole zeigt an, dass das Abonnement geändert wird.



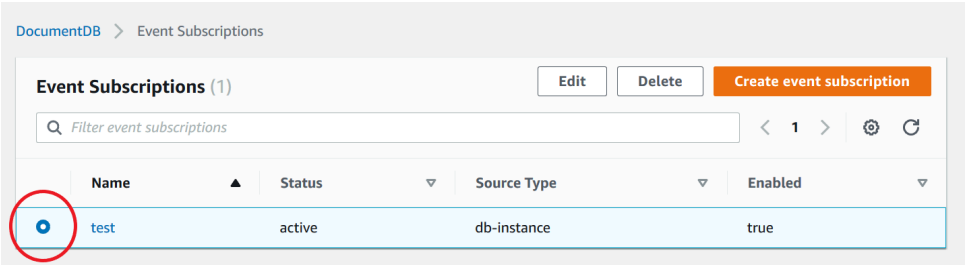
Löschen eines Abonnements für Amazon DocumentDB DocumentDB-Ereignisbenachrichtigungen

Sie können ein Abonnement löschen, wenn Sie es nicht mehr benötigen. Alle Abonnenten des Themas erhalten dann keine weiteren Ereignisbenachrichtigungen, die über dieses Abonnement ausgegeben wurden.

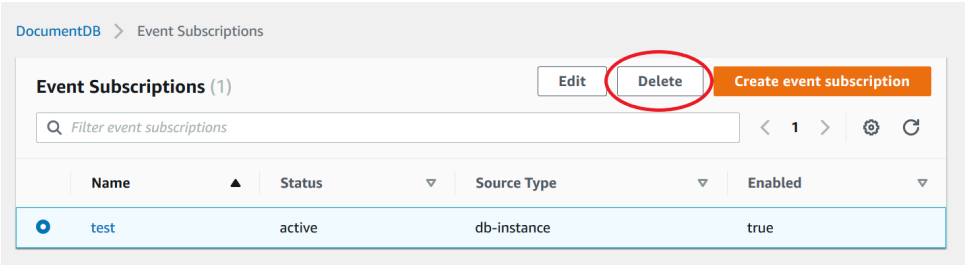
1. Melden Sie sich AWS-Managementkonsole unter an. https://console.aws.amazon.com/docdb
2. Wählen Sie im Navigationsbereich Ereignisabonnements aus.



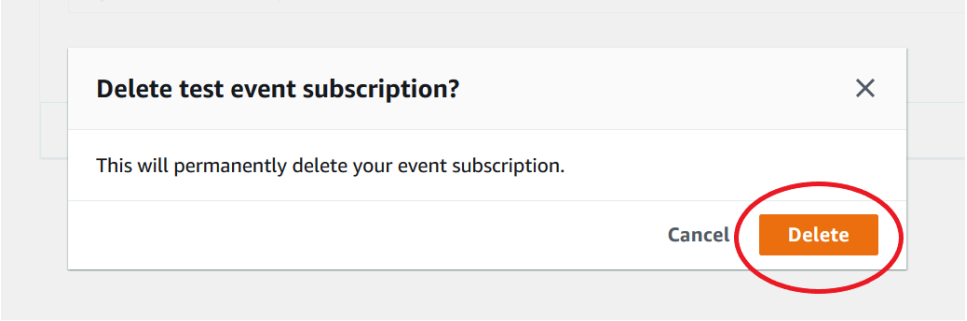
3. Wählen Sie im Bereich Ereignisabonnements das Abonnement aus, das Sie löschen möchten.



4. Wählen Sie Löschen aus.



5. In einem Popup-Fenster werden Sie gefragt, ob Sie diese Benachrichtigung dauerhaft löschen möchten. Wählen Sie Löschen aus.



Amazon DocumentDB DocumentDB-Ereigniskategorien und Meldungen

Amazon DocumentDB generiert eine beträchtliche Anzahl von Ereignissen in Kategorien, die Sie über die Konsole abonnieren können. Jede Kategorie bezieht sich auf einen Quelltyp, bei dem es sich um eine Instance, einen Cluster, einen Snapshot oder eine Parametergruppe handeln kann.

> **Note**
>
> Amazon DocumentDB verwendet bestehende Amazon RDS-Ereignisdefinitionen und -IDs.

Amazon DocumentDB DocumentDB-Ereignisse, die von Instances stammen

Kategorie	Description
Verfügbarkeit	Die Instanz wurde neu gestartet.
Verfügbarkeit	Die Instanz wurde heruntergefahren.
Konfigurationsänderung	Änderung auf eine Instanzklasse anwenden.
Konfigurationsänderung	Das Anwenden der Änderung auf eine Instanzklasse wurde abgeschlossen.
Konfigurationsänderung	Primäre Anmeldeinformationen zurücksetzen.
Erstellung	Instanz wurde erstellt.
Löschung	Instanz wurde gelöscht
Ausfall	Die Instanz ist aufgrund einer inkompatiblen Konfiguration oder eines zugrunde liegenden Speicherproblems ausgefallen. Starten Sie eine Point-in-Time-Wiederherstellung für die Instanz.
Benachrichtigung	Die Instanz wurde gestoppt.
Benachrichtigung	Instanz wurde gestartet.

Kategorie	Description
Benachrichtigung	Die Instanz wird gestartet, weil sie die maximal zulässige Zeit für das Stoppen überschritten hat.
Wiederherstellung	Die Wiederherstellung der Instanz wurde gestartet. Die Wiederherstellungsdauer variiert je nach zu wiederherstellender Datenmenge.
Wiederherstellung	Die Wiederherstellung der Instanz ist abgeschlossen.
Ausführen von Sicherheits-Patches	Das Betriebssystem-Update ist für Ihre Instance verfügbar. Informationen zum Anwenden von Aktualisierungen finden Sie unter Wartung von Amazon DocumentDB.

Amazon DocumentDB DocumentDB-Ereignisse, die aus einem Cluster stammen

Kategorie	Description
Erstellung	Cluster wurde erstellt
Löschung	Cluster wurde gelöscht.
Failover	Der vorherige Primärserver wird erneut beworben.
Failover	Der Failover zur Instanz wurde abgeschlossen.
Failover	Der Failover zur DB-Instance wurde gestartet: %s
Failover	Derselbe AZ-Failover zur DB-Instance wurde gestartet: %s

Kategorie	Description
Failover	Cross-AZ-Failover zur DB-Instance gestartet: %s
Wartung	Der Cluster wurde gepatcht.
Wartung	Der Datenbank-Cluster befindet sich in einem Zustand, der nicht aktualisiert werden kann: %s
Benachrichtigung	Der Cluster wurde gestoppt.
Benachrichtigung	Der Cluster wurde gestartet.
Benachrichtigung	Der Clusterstopp ist fehlgeschlagen.
Benachrichtigung	Der Cluster wird gestartet, weil er die maximal zulässige Zeit für das Stoppen überschritten hat.
Benachrichtigung	Der Cluster wurde von %s in %s umbenannt.

Amazon DocumentDB DocumentDB-Ereignisse, die aus einem Cluster-Snapshot stammen

Die folgende Tabelle zeigt die Ereigniskategorie und eine Liste der Ereignisse, wenn ein Amazon DocumentDB-Cluster-Snapshot der Quelltyp ist.

Kategorie	Description
backup	Ein manueller Cluster-Snapshot wird erstellt.
backup	Ein manueller Cluster-Snapshot wurde erstellt.
backup	Erstellen eines automatisierten Cluster-S napshots.
backup	Automatisierter Cluster-Snapshot erstellt.

Amazon DocumentDB DocumentDB-Ereignisse, die aus einer Parametergruppe stammen

Die folgende Tabelle zeigt den Ereignistyp sowie eine Liste der Ereignisse für den Fall, dass der Quelltyp „Parametergruppe“ ist.

Kategorie	Description
Konfigurationsänderung	Der Parameter %s wurde mit der Methode %s auf %s aktualisiert

Überwachung von Amazon DocumentDB mit CloudWatch

Amazon DocumentDB (mit MongoDB-Kompatibilität) ist in Amazon integriert, CloudWatch sodass Sie Betriebsmetriken für Ihre Cluster sammeln und analysieren können. Sie können diese Metriken mithilfe der CloudWatch Konsole, der Amazon DocumentDB-Konsole, der AWS Command Line Interface (AWS CLI) oder der API überwachen. CloudWatch

CloudWatch ermöglicht es Ihnen auch, Alarme einzurichten, sodass Sie benachrichtigt werden können, wenn ein Metrikwert einen von Ihnen angegebenen Schwellenwert überschreitet. Sie können Amazon CloudWatch Events sogar so einrichten, dass im Falle eines Verstoßes Abhilfemaßnahmen ergriffen werden. Weitere Informationen zur Verwendung CloudWatch und zu Alarmen finden Sie in der CloudWatch Amazon-Dokumentation.

Themen

- Amazon DocumentDB-Metriken
- CloudWatch Daten anzeigen
- Abmessungen von Amazon DocumentDB
- Überwachung der Opcounter-Metriken
- Überwachen von Datenbankverbindungen

Amazon DocumentDB-Metriken

Um den Zustand und die Leistung Ihres Amazon DocumentDB-Clusters und Ihrer Instances zu überwachen, können Sie die folgenden Metriken in der Amazon DocumentDB-Konsole einsehen.

> **Note**
>
> Die Metriken in den folgenden Tabellen gelten sowohl für instanzbasierte als auch für elastische Cluster.

Themen

- Metriken zur Ressourcenauslastung
- Latenzmetriken
- NVMe-backed Metriken für Instanzen
- Metriken zum Betrieb
- Command-level Metriken
- Durchsatzmetriken
- Systemmetriken
- Metriken der T3-Instanz

Metriken zur Ressourcenauslastung

Metrik	Description	
`BackupRetentionPeriodStorageUsed`	Die Gesamtmenge des Backup-Speichers in Byte, der zur Unterstützung der Point-in-Time-Wiederherstellungsfunktion innerhalb des Aufbewahrungsfensters von Amazon DocumentDB verwendet wird. Ist in dem von der `TotalBackupStorageBilled` -Metrik gemeldeten Gesamtwert enthalten. Wird für jeden Amazon DocumentDB-Cluster separat berechnet.	

Metrik	Description	
`ChangeStreamLogSize`	Die Menge des Speichers, der vom Cluster zum Speichern des Änderungsstream-Protokolls in Megabyte verwendet wird. Dieser Wert ist eine Teilmenge des Gesamtspeichers für den Cluster (`VolumeBytesUsed`) und wirkt sich auf die Kosten des Clusters aus. Preisinformationen für Speicher finden Sie auf der Amazon DocumentDB-Produktseite. Die Protokollgröße des Änderungsstreams ist vom Umfang der Änderungen auf Ihrem Cluster und der Aufbewahrungsdauer des Änderungsstream-Protokoll abhängig. Weitere Informationen zu Änderungsstreams finden Sie unter Change Streams mit Amazon DocumentDB verwenden.	
`CPUUtilization`	Prozentsatz der CPU-Nutzung durch eine Instance.	
`DatabaseConnections`	Die Anzahl der Verbindungen (aktiv und inaktiv), die auf einer Instance im Abstand von 1 Minute geöffnet werden.	

Metrik	Description
DatabaseConnectionsMax	Die maximale Anzahl offener Datenbankverbindungen (aktiv und inaktiv) auf einer Instance in einem Zeitraum von 1 Minute.
DatabaseConnectionsLimit	Die maximale Anzahl gleichzeitiger Datenbankverbindungen (aktiv und inaktiv), die auf einer Instance zu einem bestimmten Zeitpunkt zulässig sind.
DatabaseConnectionsActiveMax	Die maximale Anzahl aktiver Datenbankverbindungen auf einer Instance in einem Zeitraum von 1 Minute. Aktive Verbindungen sind diejenigen, die gerade einen Vorgang ausführen, im Gegensatz zu inaktiven Verbindungen. Die Kontingente für aktive Verbindungen pro Instanztyp finden Sie unterKontingente für Instanzen.
DatabaseCursors	Die Anzahl der Cursor, die auf einer Instance geöffnet werden, wird im Abstand von 1 Minute ausgeführt.
DatabaseCursorsMax	Die maximale Anzahl offener Cursor auf einer Instanz in einem Zeitraum von 1 Minute.

Metrik	Description
DatabaseCursorsLimit	Die maximale Anzahl von Cursorn, die zu einem bestimmten Zeitpunkt auf einer Instanz zulässig sind.
DatabaseCursorsTimedOut	Die Anzahl der Cursor, bei denen innerhalb eines Zeitraums von 1 Minute ein Timeout aufgetreten ist.
FreeableMemory	Die Menge des verfügbaren Arbeitsspeichers (RAM-Speicher) in Bytes.
FreeLocalStorage	Diese Metrik gibt die Menge an Speicher an, der für jede Instance für temporäre Tabellen und Protokolle zur Verfügung steht. Dieser Wert hängt von der Instance-Klasse ab. Sie können die Menge an freiem Speicherplatz für eine Instance erhöhen, indem Sie eine größere Instance-Klasse für Ihre Instance auswählen. (Dies gilt nicht für DocumentDB Serverless.)
LowMemThrottleQueueDepth	Die Warteschlangentiefe für Anfragen, die aufgrund des geringen verfügbaren Speichers gedrosselt werden, wird im Abstand von 1 Minute abgerufen.

Metrik	Description
`LowMemThrottleMaxQueueDepth`	Die maximale Warteschlangentiefe für Anfragen, die aufgrund von zu wenig verfügbarem Speicher gedrosselt werden, in einem Zeitraum von 1 Minute.
`LowMemNumOperationsThrottled`	Die Anzahl der Anfragen, die in einem Zeitraum von 1 Minute aufgrund von zu wenig verfügbarem Speicher gedrosselt werden.
`SnapshotStorageUsed`	Die Gesamtmenge an Backup-Speicherplatz in Byte, die von allen Snapshots für einen bestimmten Amazon DocumentDB-Cluster außerhalb seines Aufbewahrungsfensters für Backups verbraucht wird. Ist in dem von der `TotalBackupStorageBilled` -Metrik gemeldeten Gesamtwert enthalten. Wird für jeden Amazon DocumentDB-Cluster separat berechnet.
`SwapUsage`	Größe des auf der Instance genutzten Auslagerungsbereichs.

Metrik	Description
`TotalBackupStorageBilled`	Die Gesamtmenge an Backup-Speicherplatz in Byte, für die Ihnen ein bestimmtes Amazon DocumentDB-Cluster in Rechnung gestellt wird. Umfasst den Sicherungsspeicher gemessen an den Metriken `SnapshotStorageUsed` und `BackupRetentionPeriodStorageUsed`. Wird für jeden Amazon DocumentDB-Cluster separat berechnet.
`TransactionsOpen`	Die Anzahl der auf einer Instance geöffneten Transaktionen, die im Abstand von 1 Minute durchgeführt werden.
`TransactionsOpenMax`	Die maximale Anzahl von Transaktionen, die auf einer Instance innerhalb eines Zeitraums von 1 Minute geöffnet werden.
`TransactionsOpenLimit`	Die maximale Anzahl gleichzeitiger Transaktionen, die zu einem bestimmten Zeitpunkt auf einer Instance zulässig sind.

Metrik	Description
VolumeBytesUsed	Die Menge des von Ihrem Cluster verwendeten Speicherplatzes in Byte. Dieser Wert wirkt sich auf die Kosten des Clusters aus. Preisinformationen finden Sie auf der Amazon DocumentDB-Produktseite.

Latenzmetriken

Metrik	Description
DBClusterReplicaLagMaximum	Die maximale Verzögerung in Millisekunden zwischen der primären Instance und jeder Amazon DocumentDB-Instance im Cluster.
DBClusterReplicaLagMinimum	Die Mindestverzögerung zwischen der Primär-Instance und jeder Replikat-Instance im Cluster, angegeben in Millisekunden.
DBInstanceReplicaLag	Die Verzögerung in Millisekunden, wenn Updates von der Primär-Instance an eine Replikat-Instance repliziert werden.
ReadLatency	Die durchschnittliche Zeit, die pro Festplattenvorgang benötigt wird. I/O

Metrik	Description
`WriteLatency`	Die durchschnittliche Zeit, in Millisekunden, die pro Festplattenvorgang benötigt wird. I/O

NVMe-backed Metriken für Instanzen

Metrik	Description
`NVMeStorageCacheHitRatio`	Der Prozentsatz der Anfragen, die vom mehrstufigen Cache bedient werden.
`FreeNVMeStorage`	Die Menge des verfügbaren flüchtigen NVMe-Speicherplatzes
`ReadIOPSNVMeStorage`	Die durchschnittliche Anzahl von I/O Festplattenlesevorgängen im temporären NVMe-Speicher.
`ReadLatencyNVMeStorage`	Die durchschnittliche Zeit, die pro Lesevorgang auf der Festplatte für temporären I/O NVMe-Speicher benötigt wird.
`ReadThroughputNVMeStorage`	Die durchschnittliche Anzahl von Bytes, die pro Sekunde von der Festplatte für den flüchtigen NVMe-Speicher gelesen werden.
`WriteIOPSNVMeStorage`	Die durchschnittliche Anzahl von I/O Festplattenschreib

Metrik	Description
	vorgängen in temporären NVMe-Speicher.
`WriteLatencyNVMeStorage`	Die durchschnittliche Zeit, die pro I/O Festplattenschreibvorgang für temporären NVMe-Speicher benötigt wird.
`WriteThroughputNVMeStorage`	Die durchschnittliche Anzahl von Bytes, die pro Sekunde für den flüchtigen NVMe-Speicher auf die Festplatte geschrieben werden

Metriken zum Betrieb

Metrik	Description
`DocumentsDeleted`	Die Anzahl gelöschter Dokumente in einem Zeitraum von 1 Minute.
`DocumentsInserted`	Die Anzahl der eingefügten Dokumente in einem Zeitraum von 1 Minute.
`DocumentsReturned`	Die Anzahl der zurückgegebenen Dokumente in einem Zeitraum von 1 Minute.
`DocumentsUpdated`	Die Anzahl der aktualisierten Dokumente in einem Zeitraum von 1 Minute.

Metrik	Description
OpcountersCommand	Die Anzahl der Befehle, die in einem Zeitraum von 1 Minute ausgegeben wurden.
OpcountersDelete	Die Anzahl der Löschvorgänge, die innerhalb eines Zeitraums von 1 Minute ausgeführt wurden.
OpcountersGetmore	Die Anzahl der Getmores, die innerhalb eines Zeitraums von 1 Minute ausgegeben wurden.
OpcountersInsert	Die Anzahl der Insert-Operationen, die innerhalb eines Zeitraums von 1 Minute ausgeführt wurden.
OpcountersQuery	Die Anzahl der Abfragen, die innerhalb eines Zeitraums von 1 Minute gestellt wurden.
OpcountersUpdate	Die Anzahl der Aktualisierungsvorgänge, die innerhalb eines Zeitraums von 1 Minute ausgeführt wurden.
TransactionsStarted	Die Anzahl der Transaktionen, die in einem Zeitraum von 1 Minute auf einer Instance gestartet wurden.
TransactionsCommitted	Die Anzahl der Transaktionen, die auf einer Instance innerhalb eines Zeitraums von 1 Minute ausgeführt wurden.

Metrik	Description	
`TransactionsAborted`	Die Anzahl der Transakti onen, die auf einer Instance innerhalb eines Zeitraums von 1 Minute abgebrochen wurden.	
`TTLDeletedDocuments`	Die Anzahl der Dokumente, die von einem TTLMonitor in einem Zeitraum von 1 Minute gelöscht wurden.	

Command-level Metriken

Amazon DocumentDB veröffentlicht Latenz- und Parallelitätsmetriken pro Befehl für. CloudWatch Mithilfe dieser Metriken können Sie ermitteln, welche Vorgänge am meisten zu Ihrer Arbeitslast beitragen, und Probleme mit der Endlatenz oder dem Druck auf Parallelität erkennen.

Jeder Befehl in der folgenden Tabelle gibt drei Metriken in einem Zeitraum von 1 Minute aus:

- `AvgDuration`— Die durchschnittliche Ausführungszeit des Befehls in Mikrosekunden.
- `P100Duration`— Die maximale Ausführungszeit des Befehls (100. Perzentil) in Mikrosekunden.
- `MaxConcurrent`— Die maximale Anzahl gleichzeitiger Ausführungen des Befehls.

Metriknamen folgen dem Muster. `Command.`*`commandName`*`.`*`metricName`* Zum Beispiel `Command.find.AvgDuration` oder `Command.aggregate.MaxConcurrent`.

Metrik	Description
`Command.find`	Lesen Sie Abfragen mithilfe von find.
`Command.insert`	Operationen zum Einfügen von Dokumenten.
`Command.findAndMod ify`	Atomare Operationen zum Suchen und Ändern.

Metrik	Description
`Command.update`	Operationen zur Aktualisierung von Dokumenten.
`Command.delete`	Vorgänge zum Löschen von Dokumenten.
`Command.aggregate`	Ausführungen von Aggregationspipelines.
`Command.count`	Operationen zählen.
`Command.distinct`	Abfragen mit unterschiedlichen Werten.
`Command.getMore`	Operationen zum Abrufen von Cursor-Batches.
`Command.abortTransaction`	Operationen zum Abbrechen von Transaktionen.
`Command.commitTransaction`	Operationen zum Festschreiben von Transaktionen.

Durchsatzmetriken

Metrik	Description
`NetworkReceiveThroughput`	Der von Clients erhaltene Netzwerkdurchsatz für jede Instance im Cluster, angegeben in Bytes pro Sekunde. Dieser Durchsatz beinhaltet nicht den Netzwerkdatenverkehr zwischen den Instances im Cluster und dem Cluster-Volume.
`NetworkThroughput`	Die Menge des Netzwerkdurchsatzes in Byte pro Sekunde, der von jeder Instance im Amazon

Metrik	Description	
	DocumentDB-Cluster sowohl empfangen als auch an Clients übertragen wird. Dieser Durchsatz beinhaltet nicht den Netzwerkdatenverkehr zwischen den Instances im Cluster und dem Cluster-Volume.	
`NetworkTransmitThroughput`	Der an Clients gesendete Netzwerkdurchsatz für jede Instance im Cluster, angegeben in Bytes pro Sekunde. Dieser Durchsatz beinhaltet nicht den Netzwerkdatenverkehr zwischen den Instances im Cluster und dem Cluster-Volume.	
`ReadIOPS`	Die durchschnittliche Anzahl von I/O Festplattenlesevorgängen pro Sekunde. Amazon DocumentDB meldet Lese- und Schreib-IOPS getrennt und in Intervallen von 1 Minute.	
`ReadThroughput`	Die durchschnittliche Anzahl Byte, die pro Sekunde vom Datenträger gelesen werden.	

Metrik	Description
StorageNetworkReceiveThroughput	Der Netzwerkdurchsatz, in Byte pro Sekunde, der von jeder Instance im Cluster vom Amazon DocumentDB-Cluster-Speichervolume empfangen wird.
StorageNetworkTransmitThroughput	Die Menge an Netzwerkdurchsatz in Byte pro Sekunde, die von jeder Instance im Cluster an das Amazon DocumentDB-Cluster-Speichervolume gesendet wird.
StorageNetworkThroughput	Der Netzwerkdurchsatz in Byte pro Sekunde, der von jeder Instance im Amazon DocumentDB-Cluster empfangen und an das Amazon DocumentDB-Cluster-Speichervolume gesendet wurde.

Metrik	Description
`VolumeReadIOPs`	Die durchschnittliche Anzahl der abgerechneten I/O Lesevorgänge von einem Cluster-Volume, die in Intervallen von 5 Minuten gemeldet wird. In Rechnung gestellte Lesevorgänge werden auf Cluster-Volume-Ebene berechnet, aus allen Instances im Cluster zusammengestellt und dann in fünfminütigen Intervallen gemeldet. Der Wert wird anhand des Werts der Metrik für Leseoperationen über einen fünfminütigen Zeitraum berechnet. Sie können die Menge an in Rechnung gestellten Leseoperationen pro Sekunde bestimmen, indem Sie den Wert der Metrik für in Rechnung gestellte Leseoperationen durch 300 Sekunden teilen. Wenn beispielsweise 13.686 `VolumeReadIOPs` zurückgegeben wird, sind die fakturierten Lesevorgänge pro Sekunde 45 (13.686/300 = 45,62). In Rechnung gestellte Operationen fallen für Abfragen für nicht im Buffer-

Metrik	Description	
	Cache enthaltene Datenbank seiten an, die erst aus dem Speicher geladen werden müssen. Sie sehen evtl. Spitzenwerte in den in Rechnung gestellten Operationen, da Abfrageergebnisse aus dem Speicher gelesen und anschließend in den Buffer-Cache geladen werden.	

Metrik	Description	
`VolumeWriteIOPs`	Die durchschnittliche Anzahl der abgerechneten I/O Schreibvorgänge von einem Cluster-Volume, die in Intervallen von 5 Minuten gemeldet wird. In Rechnung gestellte Schreibvorgänge werden auf Cluster-Volume-Ebene berechnet, aus allen Instances im Cluster aggregiert und dann in 5-Minuten-Intervallen gemeldet. Der Wert wird anhand des Werts der Schreiboperationsmetrik über einen Zeitraum von 5 Minuten berechnet. Sie können die Menge der in Rechnung gestellten Schreiboperationen pro Sekunde ermitteln, indem Sie den Wert der Schreiboperations-Abrechnungsmetrik durch 300 Sekunden teilen. Wenn beispielsweise 13.686 `VolumeWriteIOPs` zurückgegeben wird, sind die fakturierten Schreibvorgänge pro Sekunde 45 (13.686/300 = 45,62). Beachten Sie, dass die `VolumeWriteIOPs` AND-Metriken von der DocumentDB-Speicherschicht berechnet werden `VolumeReadIOPs`	

Metrik	Description	
	und dass sie IOs einschließen, die von der Primär- und der Replikatinstanz ausgeführt werden. Die Daten werden alle 20 bis 30 Minuten aggregier t und dann in Intervallen von 5 Minuten gemeldet, sodass der gleiche Datenpunkt für die Metrik in dem Zeitraum ausgegeben wird. Wenn Sie nach einer Metrik suchen, die Ihren Einfügeoperationen über ein Intervall von 1 Minute entspricht, können Sie die WriteIOps-Metrik auf Instance-Ebene verwenden. Die Metrik ist auf der Registerkarte „Überwachung“ Ihrer primären Amazon DocumentDB-Instanc e verfügbar.	
`WriteIOPS`	Die durchschnittliche Anzahl von I/O Festplatten-Schrei bvorgängen pro Sekunde. Wenn sie auf Clustereb ene verwendet `WriteIOPs` werden, werden sie für alle Instanzen im Cluster ausgewertet. Lese- und Schreib-IOPS werden separat und in 1-Minuten-Intervallen angegeben.	

Metrik	Description	
`WriteThroughput`	Die durchschnittliche Anzahl von Bytes, die pro Sekunde auf den Datenträger geschrieben werden.	

Systemmetriken

Metrik	Description	
`AvailableMVCCIds`	Ein Zähler, der die Anzahl der verbleibenden verfügbaren Schreibvorgänge anzeigt, bevor Null erreicht wird. Wenn dieser Zähler Null erreicht, wechselt Ihr Cluster in den schreibgeschützten Modus, bis die IDs abgerufen und wiederverwendet werden. Der Zähler sinkt mit jedem Schreibvorgang und steigt, wenn die Garbage Collection alte MVCC-IDs wiederverwendet.	
`BufferCacheHitRatio`	Der Prozentsatz der vom Buffer-Cache bedienten Anfragen.	
`DiskQueueDepth`	Die Anzahl der I/O Operationen, die darauf warten, auf die Festplatte geschrieben oder von der Festplatte gelesen zu werden.	

Metrik	Description
`EngineUptime`	Die Gesamtlaufzeit der Instance in Sekunden.
`IndexBufferCacheHitRatio`	Der Prozentsatz der Indexanforderungen, die vom Puffercache bedient werden. Möglicherweise stellen Sie unmittelbar nach dem Löschen eines Indexes, einer Sammlung oder Datenbank einen Anstieg von mehr als 100 Prozent für die Metrik fest. Dies wird nach 60 Sekunden automatisch korrigiert. Diese Einschränkung wird in einem zukünftigen Patch-Update behoben.
`LongestActiveGCRuntime`	Dauer des längsten aktiven Garbage-Collection-Vorgangs in Sekunden. Wird jede Minute aktualisiert und verfolgt nur aktive Vorgänge, ausgenommen Prozesse, die innerhalb des einminütigen Fensters abgeschlossen werden.

Metriken der T3-Instanz

Metrik	Description
`CPUCreditUsage`	Die Anzahl der CPU-Credits, die während des Messzeitraums ausgegeben wurden.

Metrik	Description
`CPUCreditBalance`	Die Anzahl der CPU-Credits, die eine Instance angehäuft hat. Dieser Saldo wird während der Steigerung der CPU-Leistung aufgebrau cht, da die CPU-Guthaben schneller verbraucht als erworben werden.
`CPUSurplusCreditBa lance`	Die Anzahl der überschüs sigen CPU-Credits, die zur Aufrechterhaltung der CPU-Leistung ausgegeben werden, wenn der CPUCreditBalance Wert Null ist.
`CPUSurplusCreditsC harged`	Die Anzahl der überschüs sigen CPU-Credits, die die maximale Anzahl an CPU-Credits überschreiten, die innerhalb von 24 Stunden verdient werden können, und für die eine zusätzlic he Gebühr anfällt. Weitere Informationen finden Sie unter CPU-Guthaben überwachen.

CloudWatch Daten anzeigen

Sie können CloudWatch Amazon-Daten mithilfe der CloudWatch Konsole, der Amazon DocumentDB-Konsole AWS Command Line Interface (AWS CLI) oder der CloudWatch API anzeigen.

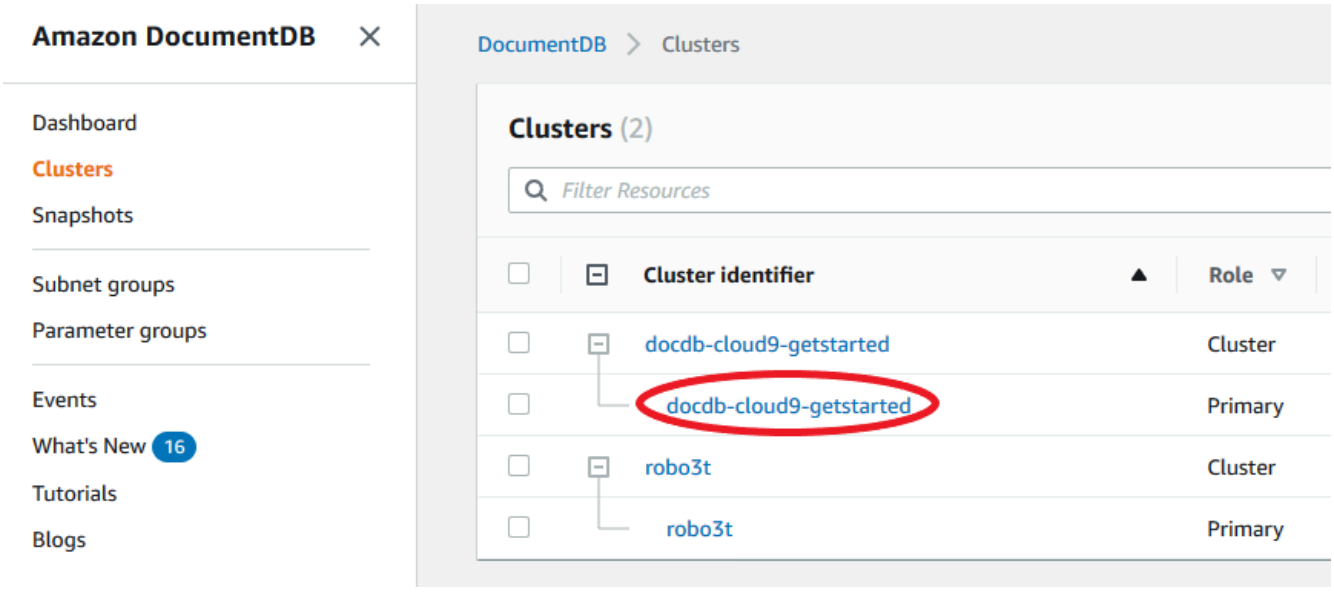
Using the AWS-Managementkonsole

Gehen Sie wie folgt vor, um CloudWatch Metriken mithilfe der Amazon DocumentDB Management Console anzuzeigen.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Klicken Sie im Navigationsbereich auf Cluster.

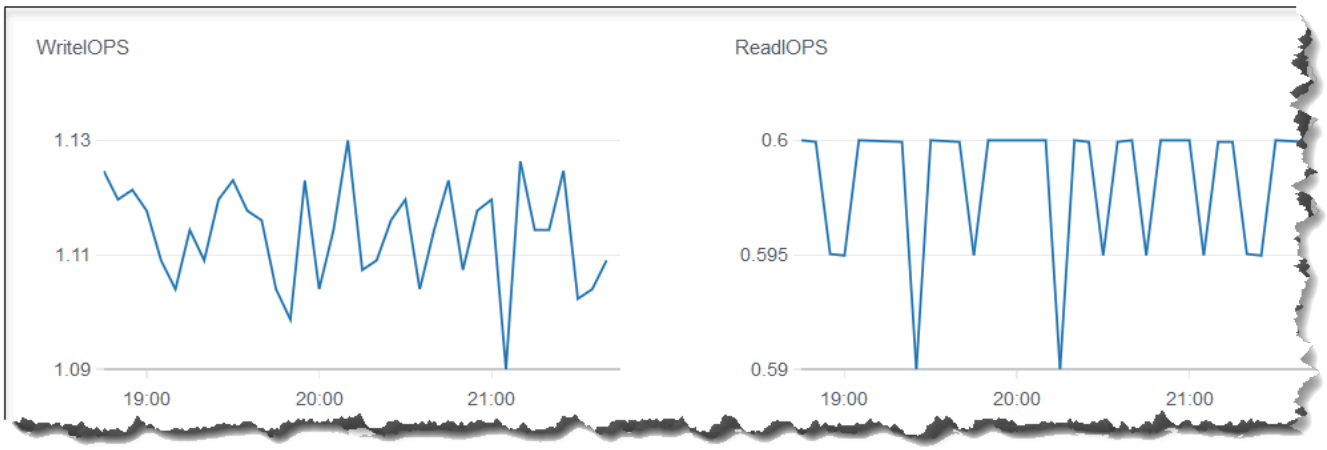
> **Tip**
>
> Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Im Navigationsfeld „Cluster“ sehen Sie die Spalte Cluster Identifier. Ihre Instances sind unter Clustern aufgeführt, ähnlich wie im folgenden Screenshot.



4. Wählen Sie aus der Liste der Instanzen den Namen der Instanz aus, für die Sie Metriken benötigen.
5. Wählen Sie auf der daraufhin angezeigten Seite mit der Instanzübersicht den Tab Monitoring, um grafische Darstellungen der Metriken Ihrer Amazon DocumentDB-Instance anzuzeigen. Da für jede Metrik ein Diagramm generiert werden muss, kann es einige Minuten dauern, bis die CloudWatch Grafiken gefüllt sind.

Die folgende Abbildung zeigt die grafischen Darstellungen von zwei CloudWatch Metriken in der Amazon DocumentDB-Konsole und`WriteIOPS`. `ReadIOPS`



Using the CloudWatch Management Console

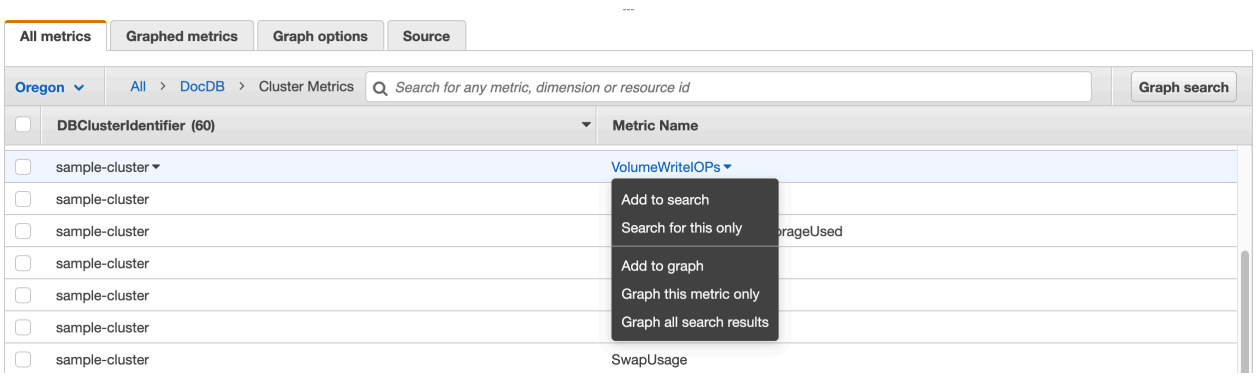
Gehen Sie wie folgt vor, um CloudWatch Metriken mithilfe der CloudWatch Management Console anzuzeigen.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unterhttps://console.aws.amazon.com/cloudwatch.
2. Wählen Sie im Navigationsbereich Metriken aus. Wählen Sie dann aus der Liste der Servicenamen DocDB aus.
3. Wählen Sie eine metrische Dimension (z. B. Cluster-Metriken).
4. Auf der Registerkarte Alle Metriken werden alle Metriken für diese Dimension in DocDB angezeigt.

    a. Um die Tabelle sortieren, verwenden Sie die Spaltenüberschrift.

    b. Um eine Metrik grafisch darzustellen, müssen Sie das Kontrollkästchen neben der Metrik aktivieren. Um alle Metriken auszuwählen, aktivieren Sie das Kontrollkästchen in der Kopfzeile der Tabelle.

    c. Um nach Metrik zu filtern, bewegen Sie den Mauszeiger über den Metriknamen und wählen Sie den Dropdown-Pfeil neben dem Metriknamen aus. Wählen Sie dann Zur Suche hinzufügen aus, wie in der folgenden Abbildung gezeigt.



Using the AWS CLI

Um CloudWatch Daten für Amazon DocumentDB anzuzeigen, verwenden Sie den CloudWatch `get-metric-statistics` Vorgang mit den folgenden Parametern.

Parameters

- **`--namespace`** – Erforderlich. Der Service-Namespace, für den Sie CloudWatch -Metriken anzeigen möchten. Für Amazon DocumentDB muss dies der Fall sein. `AWS/DocDB`
- **`--metric-name`** – Erforderlich. Der Name der Metrik, für die Sie Daten benötigen.
- **`--start-time`** – Erforderlich. Der Zeitstempel, der den ersten zurückzugebenden Datenpunkt bestimmt.

  Der angegebene Wert wird eingeschlossen. Die Ergebnisse umfassen Datenpunkte mit dem angegebenen Zeitstempel. Der Zeitstempel muss im UTC-Format nach ISO 8601 vorliegen (z. B. T23:00:00Z 2016-10-03).
- **`--end-time`** – Erforderlich. Der Zeitstempel, der den letzten zurückzugebenden Datenpunkt bestimmt.

  Der angegebene Wert wird eingeschlossen. Die Ergebnisse umfassen Datenpunkte mit dem angegebenen Zeitstempel. Der Zeitstempel muss im UTC-Format ISO 8601 vorliegen (z. B. 2016-10-03). T23:00:00Z
- **`--period`** – Erforderlich. Die Granularität der zurückgegebenen Datenpunkte in Sekunden. Für Metriken mit regulärer Auflösung kann ein Zeitraum gerade mal eine Minute (60 Sekunden) betragen. Der Wert muss ein Vielfaches von 60 sein. Für hochauflösende Metriken, die in Abständen von weniger als einer Minute erfasst werden, kann der Zeitraum 1, 5, 10, 30, 60 oder ein Vielfaches von 60 betragen.

- **`--dimensions`**— Fakultativ. Wenn die Metrik mehrere Dimensionen enthält, müssen Sie für jede Dimension einen Wert angeben. CloudWatch behandelt jede eindeutige Kombination von Dimensionen als separate Metrik. Wenn eine bestimmte Kombination von Dimensionen nicht veröffentlicht wurde, können Sie keine Metriken dafür abrufen. Sie müssen die gleichen Dimensionen angeben, die bei der Erstellung der Metriken verwendet wurden.
- **`--statistics`**— Fakultativ. Die Metrik-Statistiken abgesehen von den Perzentil-Statistiken. Verwenden Sie für Perzentil-Statistiken `ExtendedStatistics`. Wenn Sie `GetMetricStatistics` aufrufen, müssen Sie `Statistics` oder `ExtendedStatistics` angeben, aber nicht beides.

  Gültige Werte:

  - `SampleCount`
  - `Average`
  - `Sum`
  - `Minimum`
  - `Maximum`
- **`--extended-statistics`**— Fakultativ. Die `percentile`-Statistiken. Geben Sie Werte zwischen p0.0 und p100 an. Wenn Sie `GetMetricStatistics` aufrufen, müssen Sie `Statistics` oder `ExtendedStatistics` angeben, aber nicht beides.
- **`--unit`**— Fakultativ. Die Einheit für eine bestimmte Metrik. Metriken können in mehreren Einheiten angegeben werden. Wenn Sie keine Einheit festlegen, werden alle Einheiten zurückgegeben. Wenn Sie nur eine Einheit angeben, die für die Metrik nicht unterstützt wird, erhalten Sie keine Ergebnisse für den Aufruf.

  Mögliche Werte:

  - `Seconds`
  - `Microseconds`
  - `Milliseconds`
  - `Bytes`
  - `Kilobytes`
  - `Megabytes`
  - `Gigabytes`
  - `Terabytes`

- Bits
- Kilobytes
- Megabits
- Gigabits
- Terabits
- Percent
- Count
- Bytes/Second
- Kilobytes/Second
- Megabytes/Second
- Gigabytes/Second
- Terabytes/Second
- Bits/Second
- Kilobits/Second
- Megabits/Second
- Gigabits/Second
- Terabits/Second
- Count/Second
- None

Example

Im folgenden Beispiel wird die maximale CPUUtilization für einen Zeitraum von 2 Stunden gezeigt, wobei alle 60 Sekunden eine Messung erfolgt.

Für Linux, macOS oder Unix:

```
aws cloudwatch get-metric-statistics \
       --namespace AWS/DocDB \
       --dimensions \
           Name=DBInstanceIdentifier,Value=docdb-2019-01-09-23-55-38 \
       --metric-name CPUUtilization \
       --start-time 2019-02-11T05:00:00Z \
       --end-time 2019-02-11T07:00:00Z \
       --period 60 \
```

```
       --statistics Maximum
```

Für Windows:

```
aws cloudwatch get-metric-statistics ^
       --namespace AWS/DocDB ^
       --dimensions ^
           Name=DBInstanceIdentifier,Value=docdb-2019-01-09-23-55-38 ^
       --metric-name CPUUtilization ^
       --start-time 2019-02-11T05:00:00Z ^
       --end-time 2019-02-11T07:00:00Z ^
       --period 60 ^
       --statistics Maximum
```

Die Ausgabe dieser Operation sieht in etwa wie folgt aus.

```
{
       "Label": "CPUUtilization",
       "Datapoints": [
           {
               "Unit": "Percent",
               "Maximum": 4.49152542374361,
               "Timestamp": "2019-02-11T05:51:00Z"
           },
           {
               "Unit": "Percent",
               "Maximum": 4.25000000000485,
               "Timestamp": "2019-02-11T06:44:00Z"
           },

           ********* some output omitted for brevity *********

           {
               "Unit": "Percent",
               "Maximum": 4.33333333331878,
               "Timestamp": "2019-02-11T06:07:00Z"
           }
       ]
   }
```

Abmessungen von Amazon DocumentDB

Die Metriken für Amazon DocumentDB werden durch die Werte für das Konto oder den Vorgang qualifiziert. Sie können die CloudWatch Konsole verwenden, um Amazon DocumentDB-Daten abzurufen, die nach einer der Dimensionen in der folgenden Tabelle gefiltert wurden.

Dimension	Description
`DBClusterIdentifier`	Filtert die Daten, die Sie für einen bestimmten Amazon DocumentDB-Cluster anfordern.
`DBClusterIdentifier, Role`	Filtert die Daten, die Sie für einen bestimmten Amazon DocumentDB-Cluster anfordern, und aggregiert die Metrik nach Instance-Rolle (). WRITER/READER Sie können beispielsweise Metriken für alle READER-Instances eines Clusters zusammenfassen.
`DBInstanceIdentifier`	Filtert die angeforderten Daten für eine spezifische Datenbank-Instance.

Überwachung der Opcounter-Metriken

Opcounter-Metriken haben einen Wert ungleich Null (normalerweise ~50) für Cluster im Leerlauf. Das liegt daran, dass Amazon DocumentDB regelmäßige Zustandsprüfungen, interne Abläufe und Aufgaben zur Erfassung von Kennzahlen durchführt.

Überwachen von Datenbankverbindungen

Wenn Sie die Anzahl der Verbindungen mithilfe von Datenbank-Engine-Befehlen wie anzeigen`db.runCommand( { serverStatus: 1 })`, werden Ihnen möglicherweise bis zu 10 Verbindungen mehr angezeigt, als Sie in `DatabaseConnections` CloudWatch Durchsicht sehen. Dies liegt daran, dass Amazon DocumentDB regelmäßig Zustandsprüfungen und Aufgaben zur Erfassung von Kennzahlen durchführt, die in `DatabaseConnections` nicht berücksichtigt werden. `DatabaseConnections`steht nur für vom Kunden initiierte Verbindungen.

Protokollieren Amazon DocumentDB DocumentDB-API-Aufrufen mit AWS CloudTrail

Amazon DocumentDB (mit MongoDB-Kompatibilität) ist in einen Service integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen von Benutzern, Rollen oder einem AWS Service in Amazon DocumentDB (mit MongoDB-Kompatibilität) bereitstellt. CloudTrail erfasst alle AWS CLI API-Aufrufe für Amazon DocumentDB als Ereignisse, einschließlich Aufrufe von der Amazon DocumentDB-Konsole und von Codeaufrufen an das Amazon DocumentDB-SDK. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Übermittlung von CloudTrail Ereignissen an einen Amazon S3 S3-Bucket aktivieren, einschließlich Ereignissen für Amazon DocumentDB. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse trotzdem auf der CloudTrail Konsole im Event-Verlauf einsehen. Anhand der von CloudTrail gesammelten Informationen können Sie die Anfrage an Amazon DocumentDB (mit MongoDB-Kompatibilität), die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde und andere Details ermitteln.

> **Important**
>
> Für bestimmte Verwaltungsfunktionen verwendet Amazon DocumentDB eine Betriebstechnologie, die mit Amazon Relational Database Service (Amazon RDS) gemeinsam genutzt wird. Amazon DocumentDB DocumentDB-Konsolen- und API-Aufrufe werden als Aufrufe der Amazon RDS-API protokolliert. AWS CLI

Weitere Informationen AWS CloudTrail finden Sie im AWS CloudTrail Benutzerhandbuch.

Amazon DocumentDB DocumentDB-Informationen in CloudTrail

CloudTrail ist auf Ihrem aktiviert AWS-Konto , wenn Sie das Konto erstellen. Wenn Aktivitäten in Amazon DocumentDB (mit MongoDB-Kompatibilität) auftreten, wird diese Aktivität zusammen mit anderen AWS Serviceereignissen in der CloudTrail Ereignishistorie in einem Ereignis aufgezeichnet. Sie können aktuelle Ereignisse in Ihrem anzeigen, suchen und herunterladen. AWS-Konto Weitere Informationen finden Sie unter Ereignisse mit CloudTrail Ereignisverlauf anzeigen.

Für eine fortlaufende Aufzeichnung der Ereignisse in Ihrem AWS-Konto, einschließlich Ereignissen für Amazon DocumentDB (mit MongoDB-Kompatibilität), erstellen Sie einen Trail. Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Wenn Sie einen Trail in der Konsole anlegen, gilt dieser für alle AWS-Regionen-Regionen. Der Trail protokolliert

Ereignisse aus allen Regionen in der AWS -Partition und stellt die Protokolldateien in dem von Ihnen angegebenen Amazon-S3-Bucket bereit. Darüber hinaus können Sie andere AWS Dienste konfigurieren, um die in den CloudTrail Protokollen gesammelten Ereignisdaten weiter zu analysieren und darauf zu reagieren. Weitere Informationen finden Sie in folgenden Themen im AWS CloudTrail - Benutzerhandbuch:

- Übersicht zum Erstellen eines Trails
- CloudTrail Unterstützte Dienste und Integrationen
- Konfiguration von Amazon SNS SNS-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail Protokolldateien aus mehreren Regionen
- Empfangen von CloudTrail Protokolldateien von mehreren Konten

Jedes Event oder jeder Protokolleintrag enthält Informationen über den Ersteller der Anfrage. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anforderung mit Root- oder -Benutzeranmeldeinformationen ausgeführt wurde.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anfrage von einem anderen AWS Dienst gestellt wurde.

Weitere Informationen finden Sie unter CloudTrail userIdentity-Element.

Profilierung von Amazon DocumentDB-Vorgängen

Sie können den Profiler in Amazon DocumentDB (mit MongoDB-Kompatibilität) verwenden, um die Ausführungszeit und Details der auf Ihrem Cluster ausgeführten Operationen zu protokollieren. Profiler ist für die Überwachung der langsamsten Operationen in Ihrem Cluster nützlich. So können Sie die Leistung einzelner Abfragen und die allgemeine Cluster-Leistung verbessern.

Standardmäßig ist die Profiler-Funktion deaktiviert. Wenn diese Option aktiviert ist, protokolliert der Profiler Vorgänge, die länger als ein vom Kunden definierter Schwellenwert dauern (z. B. 100 ms), in Amazon Logs. CloudWatch Zu den protokollierten Details gehören der Befehl, für den das Profil erstellt wird, die Uhrzeit, die Planübersicht und die Client-Metadaten. Nachdem die Vorgänge in CloudWatch Logs protokolliert wurden, können Sie Logs Insights verwenden, um Ihre Amazon CloudWatch DocumentDB-Profilerstellungsdaten zu analysieren, zu überwachen und zu archivieren. Häufige Abfragen sind in Abschnitt Häufig gestellte Abfragen zu finden.

Wenn aktiviert, verbraucht der Profiler zusätzliche Ressourcen im Cluster. Beginnen Sie mit einem hohen Schwellenwert (z. B. 500 ms) und senken Sie den Wert schrittweise, um langsame Vorgänge zu erkennen. Ein anfänglicher Schwellenwert von 50 ms kann für Anwendungen mit hohem Durchsatz zu Leistungsproblemen im Cluster führen. Der Profiler ist auf Clusterebene aktiviert und funktioniert auf allen Instanzen und Datenbanken in einem Cluster. Amazon DocumentDB protokolliert Operationen nach bestem CloudWatch Bemühen in Amazon Logs.

Amazon DocumentDB erhebt zwar keine zusätzlichen Gebühren für die Aktivierung des Profilers, Ihnen werden jedoch die Standardtarife für die Nutzung von Logs berechnet. CloudWatch Informationen zu den Preisen für CloudWatch Logs finden Sie unter Amazon-Preise. CloudWatch

Themen

- Unterstützte Vorgänge
- Einschränkungen
- Den Amazon DocumentDB-Profiler aktivieren
- Den Amazon DocumentDB-Profiler deaktivieren
- Der Export von Profiler-Protokollen wird deaktiviert
- Zugriff auf Ihre Amazon DocumentDB-Profiler-Protokolle
- Häufig gestellte Abfragen

Unterstützte Vorgänge

Amazon DocumentDB Profiler unterstützt die folgenden Operationen:

- `aggregate`
- `count`
- `delete`
- `distinct`
- `find` (OP_QUERY und Befehl)
- `findAndModify`
- `insert`
- `update`

Einschränkungen

Der Slow Query Profiler kann Profiler-Logs nur ausgeben, wenn die gesamte Ergebnismenge der Abfrage in einen Batch passt und wenn die Ergebnismenge unter 16 MB liegt (maximale BSON-Größe). Ergebnissätze, die größer als 16 MB sind, werden automatisch in mehrere Batches aufgeteilt.

Die meisten Treiber oder Shells legen möglicherweise eine kleine Standardstapelgröße fest. Sie können die Batchgröße als Teil Ihrer Abfrage angeben. Um langsame Abfrageprotokolle zu erfassen, verwenden Sie eine Batchgröße, die die Größe Ihrer erwarteten Ergebnismenge überschreitet. Wenn Sie sich über die Größe der Ergebnismenge nicht sicher sind oder wenn sie variiert, legen Sie die Batchgröße auf eine große Zahl fest (z. B. 100.000).

Wenn Sie jedoch eine größere Batchgröße verwenden, müssen mehr Ergebnisse aus der Datenbank abgerufen werden, bevor eine Antwort an den Client gesendet wird. Bei einigen Abfragen kann dies zu längeren Verzögerungen führen, bis Sie Ergebnisse erhalten. Wenn Sie nicht vorhaben, die gesamte Ergebnismenge zu verwenden, ist es möglich, dass Sie mehr für I/Os die Verarbeitung der Abfrage ausgeben und das Ergebnis wegwerfen.

Den Amazon DocumentDB-Profiler aktivieren

Die Aktivierung des Profilers auf einem Cluster erfolgt in drei Schritten. Stellen Sie sicher, dass alle Schritte abgeschlossen sind. Andernfalls werden keine Profilerstellungsprotokolle an Logs gesendet CloudWatch . Profiler wird auf Cluster-Ebene festgelegt und auf allen Datenbanken und Instances des Clusters ausgeführt.

So aktivieren Sie den Profiler in einem Cluster

1. Da Sie eine Standard-Cluster-Parametergruppe nicht ändern können, stellen Sie sicher, dass Sie über eine verfügbare benutzerdefinierte Cluster-Parametergruppe verfügen. Weitere Informationen finden Sie unter Erstellen von Amazon DocumentDB-Cluster-Parametergruppen.
2. Ändern Sie mithilfe einer verfügbaren benutzerdefinierten Cluster-Parametergruppe die folgenden Parameter: `profiler`, `profiler_threshold_ms` und `profiler_sampling_rate`. Weitere Informationen finden Sie unter Ändern der Amazon DocumentDB-Cluster-Parametergruppen.
3. Erstellen oder ändern Sie Ihren Cluster, um die benutzerdefinierte Cluster-Parametergruppe zu verwenden und den Export von `profiler` Protokollen in Logs zu CloudWatch ermöglichen.

In den folgenden Abschnitten wird gezeigt, wie Sie diese Schritte mithilfe von AWS-Managementkonsole und AWS Command Line Interface (AWS CLI) implementieren.

Using the AWS-Managementkonsole

1. Bevor Sie beginnen, erstellen Sie einen Amazon DocumentDB-Cluster und eine benutzerdefinierte Cluster-Parametergruppe, falls Sie noch keine haben. Weitere Informationen erhalten Sie unter Erstellen von Amazon DocumentDB-Cluster-Parametergruppen und Erstellen eines Amazon DocumentDB-Clusters.
2. Ändern Sie die folgenden Parameter mithilfe einer verfügbaren benutzerdefinierten Cluster-Parametergruppe. Weitere Informationen finden Sie unter Ändern der Amazon DocumentDB-Cluster-Parametergruppen.

   - `profiler`— Aktiviert oder deaktiviert die Erstellung von Abfrageprofilen. Zugelassene Werte sind `enabled` und `disabled`. Der Standardwert ist `disabled`. Um Profiling zu aktivieren, legen Sie den Wert auf `enabled` fest.
   - `profiler_threshold_ms`— Wenn auf gesetzt `profiler` ist, werden alle Befehle`enabled`, die länger dauern, als protokolliert wurden, `profiler_threshold_ms` angezeigt. CloudWatch Zugelassene Werte sind `[50-INT_MAX]`. Der Standardwert ist `100`.
   - `profiler_sampling_rate`— Der Anteil langsamer Operationen, für den ein Profil erstellt oder protokolliert werden sollte. Zugelassene Werte sind `[0.0-1.0]`. Der Standardwert ist `1.0`.
3. Ändern Sie Ihren Cluster so, dass er die benutzerdefinierte Cluster-Parametergruppe verwendet, und legen Sie fest, dass die Profiler-Protokollexporte auf Amazon veröffentlicht werden. CloudWatch

   a. Wählen Sie im Navigationsbereich die Option Clusters (Cluster), um die benutzerdefinierte Parametergruppe zu einem Cluster hinzuzufügen.

   b. Wählen Sie die Schaltfläche links neben dem Namen des Clusters, dem Sie die Parametergruppe zuordnen möchten. Wählen Sie Actions (Aktionen) und dann Modify (Ändern) aus, um den Cluster zu ändern.

   c. Wählen Sie unter Cluster options (Cluster-Optionen) die benutzerdefinierte Parametergruppe aus dem obigen Schritt aus, um sie Ihrem Cluster hinzuzufügen.

   d. Wählen Sie unter Protokollexporte die Profiler-Protokolle aus, die auf Amazon veröffentlicht werden sollen. CloudWatch

e. Wählen Sie Continue (Weiter) aus, um eine Übersicht Ihrer Änderungen anzuzeigen.

f. Nachdem Sie Ihre Änderungen überprüft haben, können Sie diese sofort oder während des nächsten Wartungsfensters unter Scheduling of modifications (Planen von Änderungen) anwenden.

g. Wählen Sie Modify cluster (Cluster ändern) aus, um den Cluster mit der neuen Parametergruppe zu aktualisieren.

Using the AWS CLI

Mit dem folgenden Verfahren wird der Profiler für alle unterstützten Operationen für den Cluster `sample-cluster` aktiviert.

1. Bevor Sie beginnen, stellen Sie sicher, dass Sie über eine benutzerdefinierte Clusterparametergruppe verfügen, indem Sie den folgenden Befehl ausführen und die Ausgabe für eine Clusterparametergruppe überprüfen, die nicht `default` im Namen und die Parametergruppenfamilie `docdb3.6` hat. Wenn Sie über keine nicht standardmäßige Clusterparametergruppe verfügen, siehe Erstellen von Amazon DocumentDB-Cluster-Parametergruppen.

```
aws docdb describe-db-cluster-parameter-groups \
    --query 'DBClusterParameterGroups[*].
[DBClusterParameterGroupName,DBParameterGroupFamily]'
```

   In der folgenden Ausgabe erfüllt nur `sample-parameter-group` beide Kriterien.

```
[
       [
           "default.docdb3.6",
           "docdb3.6"
       ],
       [
           "sample-parameter-group",
           "docdb3.6"
       ]
]
```

2. Ändern Sie mithilfe Ihrer benutzerdefinierten Cluster-Parametergruppe die folgenden Parameter:

- `profiler`— Aktiviert oder deaktiviert die Erstellung von Abfrageprofilen. Zugelassene Werte sind `enabled` und `disabled`. Der Standardwert ist `disabled`. Um Profiling zu aktivieren, legen Sie den Wert auf `enabled` fest.
- `profiler_threshold_ms`— Wenn diese Option auf gesetzt `profiler` ist`enabled`, dauern alle Befehle länger, als sie angemeldet `profiler_threshold_ms` sind. CloudWatch Zugelassene Werte sind `[50-INT_MAX]`. Der Standardwert ist `100`.
- `profiler_sampling_rate`— Der Anteil langsamer Operationen, für den ein Profil erstellt oder protokolliert werden sollte. Zugelassene Werte sind `[0.0-1.0]`. Der Standardwert ist `1.0`.

```
aws docdb modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name sample-parameter-group \
    --parameters
 ParameterName=profiler,ParameterValue=enabled,ApplyMethod=immediate \

 ParameterName=profiler_threshold_ms,ParameterValue=100,ApplyMethod=immediate \

 ParameterName=profiler_sampling_rate,ParameterValue=0.5,ApplyMethod=immediate
```

3. Ändern Sie Ihren Amazon DocumentDB-Cluster so, dass er die `sample-parameter-group` benutzerdefinierte Cluster-Parametergruppe aus dem vorherigen Schritt verwendet und den Parameter `--enable-cloudwatch-logs-exports` auf festlegt. `profiler`

   Der folgende Code ändert den Cluster so`sample-cluster`, dass er den `sample-parameter-group` aus dem vorherigen Schritt verwendet, und fügt die aktivierten CloudWatch Logs-Exporte `profiler` hinzu.

```
aws docdb modify-db-cluster \
       --db-cluster-identifier sample-cluster \
       --db-cluster-parameter-group-name sample-parameter-group \
       --cloudwatch-logs-export-configuration '{"EnableLogTypes":["profiler"]}'
```

   Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
    "DBCluster": {
        "AvailabilityZones": [
```

```
            "us-east-1c",
            "us-east-1b",
            "us-east-1a"
        ],
        "BackupRetentionPeriod": 1,
        "DBClusterIdentifier": "sample-cluster",
        "DBClusterParameterGroup": "sample-parameter-group",
        "DBSubnetGroup": "default",
        "Status": "available",
        "EarliestRestorableTime": "2020-04-07T02:05:12.479Z",
        "Endpoint": "sample-cluster.node.us-east-1.docdb.amazonaws.com",
        "ReaderEndpoint": "sample-cluster.node.us-east-1.docdb.amazonaws.com",
        "MultiAZ": false,
        "Engine": "docdb",
        "EngineVersion": "3.6.0",
        "LatestRestorableTime": "2020-04-08T22:08:59.317Z",
        "Port": 27017,
        "MasterUsername": "test",
        "PreferredBackupWindow": "02:00-02:30",
        "PreferredMaintenanceWindow": "tue:09:50-tue:10:20",
        "DBClusterMembers": [
            {
                "DBInstanceIdentifier": "sample-instance-1",
                "IsClusterWriter": true,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            },
            {
                "DBInstanceIdentifier": "sample-instance-2",
                "IsClusterWriter": true,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            }
        ],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abcd0123",
                "Status": "active"
            }
        ],
        "HostedZoneId": "ABCDEFGHIJKLM",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/sample-key",
        "DbClusterResourceId": "cluster-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
```

```
        "DBClusterArn": "arn:aws:rds:us-east-1:<accountID>:cluster:sample-
cluster",
        "AssociatedRoles": [],
        "ClusterCreateTime": "2020-01-10T22:13:38.261Z",
        "EnabledCloudwatchLogsExports": [
            "profiler"
        ],
        "DeletionProtection": true
    }
}
```

Den Amazon DocumentDB-Profiler deaktivieren

Um den Profiler zu deaktivieren, deaktivieren Sie sowohl den `profiler` Parameter als auch den Export von `profiler` Protokollen nach Logs. CloudWatch

Deaktivieren des Profilers

Sie können den `profiler` Parameter wie folgt deaktivieren AWS CLI, indem Sie entweder AWS-Managementkonsole oder verwenden.

Using the AWS-Managementkonsole

Das folgende Verfahren verwendet das AWS-Managementkonsole , um Amazon DocumentDB `profiler` zu deaktivieren.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Wählen Sie im Navigationsbereich Parameter groups (Parametergruppen) aus. Wählen Sie dann den Namen der Cluster-Parametergruppe aus, für die Sie den Profiler deaktivieren möchten.
3. Klicken Sie auf der Seite Cluster parameters (Clusterparameter) auf die Schaltfläche links neben dem `profiler`-Parameter und wählen Sie Edit (Bearbeiten).
4. Wählen Sie im Dialogfeld Modify profiler (Profiler ändern) `disabled` in der Liste aus.
5. Wählen Sie Modify Cluster Parameter (Cluster-Parameter ändern).

Using the AWS CLI

Um `profiler` auf einem Cluster mithilfe der AWS CLI zu deaktivieren, ändern Sie den Cluster wie folgt.

```
aws docdb modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name sample-parameter-group \
    --parameters
 ParameterName=profiler,ParameterValue=disabled,ApplyMethod=immediate
```

Der Export von Profiler-Protokollen wird deaktiviert

Sie können den Export von `profiler` Protokollen in Logs deaktivieren, CloudWatch indem Sie entweder das AWS-Managementkonsole oder verwenden AWS CLI, wie im Folgenden beschrieben.

Using the AWS-Managementkonsole

Im folgenden Verfahren wird der verwendet AWS-Managementkonsole , um den Export von Protokollen durch Amazon DocumentDB nach zu CloudWatch deaktivieren.

1. Öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Klicken Sie im Navigationsbereich auf Cluster. Wählen Sie die Schaltfläche links neben dem Namen des Clusters, für den Sie den Export von Protokollen deaktivieren möchten.
3. Wählen Sie aus dem Menü Actions (Aktionen) die Option Modify (Ändern) aus.
4. Scrollen Sie nach unten zum Abschnitt Log exports (Protokollexporte) und deaktivieren Sie Profiler logs (Profiler-Protokolle).
5. Klicken Sie auf Weiter.
6. Überprüfen Sie Ihre Änderungen und wählen Sie dann aus, wann diese Änderung auf Ihren Cluster angewendet werden soll.
   - Apply during the next scheduled maintenance window (Anwendung während des nächsten geplanten Wartungsfensters)
   - Apply immediately (Sofort anwenden)
7. Wählen Sie Cluster bearbeiten aus.

Using the AWS CLI

Der folgende Code ändert den Cluster `sample-cluster` und deaktiviert CloudWatch Profiler-Logs.

Example

Für Linux, macOS oder Unix:

```
aws docdb modify-db-cluster \
   --db-cluster-identifier sample-cluster \
   --cloudwatch-logs-export-configuration '{"DisableLogTypes":["profiler"]}'
```

Für Windows:

```
aws docdb modify-db-cluster ^
   --db-cluster-identifier sample-cluster ^
   --cloudwatch-logs-export-configuration '{"DisableLogTypes":["profiler"]}'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
{
    "DBCluster": {
        "AvailabilityZones": [
            "us-east-1c",
            "us-east-1b",
            "us-east-1a"
        ],
        "BackupRetentionPeriod": 1,
        "DBClusterIdentifier": "sample-cluster",
        "DBClusterParameterGroup": "sample-parameter-group",
        "DBSubnetGroup": "default",
        "Status": "available",
        "EarliestRestorableTime": "2020-04-08T02:05:17.266Z",
        "Endpoint": "sample-cluster.node.us-east-1.docdb.amazonaws.com",
        "ReaderEndpoint": "sample-cluster.node.us-east-1.docdb.amazonaws.com",
        "MultiAZ": false,
        "Engine": "docdb",
        "EngineVersion": "3.6.0",
        "LatestRestorableTime": "2020-04-09T05:14:44.356Z",
        "Port": 27017,
        "MasterUsername": "test",
        "PreferredBackupWindow": "02:00-02:30",
```

```
        "PreferredMaintenanceWindow": "tue:09:50-tue:10:20",
        "DBClusterMembers": [
            {
                "DBInstanceIdentifier": "sample-instance-1",
                "IsClusterWriter": true,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            },
            {
                "DBInstanceIdentifier": "sample-instance-2",
                "IsClusterWriter": true,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            }
        ],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abcd0123",
                "Status": "active"
            }
        ],
        "HostedZoneId": "ABCDEFGHIJKLM",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/sample-key",
        "DbClusterResourceId": "cluster-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
        "DBClusterArn": "arn:aws:rds:us-east-1:<accountID>:cluster:sample-cluster",
        "AssociatedRoles": [],
        "ClusterCreateTime": "2020-01-10T22:13:38.261Z",
        "DeletionProtection": true
    }
}
```


Zugriff auf Ihre Amazon DocumentDB-Profiler-Protokolle

Folgen Sie diesen Schritten, um auf Ihre Profilprotokolle bei Amazon CloudWatch zuzugreifen.

1. Öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Stellen Sie sicher, dass Sie sich in derselben Region wie Ihr Amazon DocumentDB-Cluster befinden.
3. Wählen Sie im Navigationsbereich Protokolle aus.

4. Um die Profiler-Protokolle für Ihren Cluster zu finden, wählen Sie in der Liste `/aws/docdb/yourClusterName/profiler` aus.

   Die Profilprotokolle für jede Ihrer Instances sind unter den jeweiligen Instance-Namen verfügbar.

Häufig gestellte Abfragen

Im Folgenden finden Sie einige häufige Abfragen, die Sie verwenden können, um Ihre Befehle, für die Profile erstellt wurden, zu analysieren. Weitere Informationen zu CloudWatch Logs Insights finden Sie unter Analysieren von Logdaten mit CloudWatch Logs Insights und Beispielabfragen.

Ruft die 10 langsamsten Operationen für eine bestimmte Sammlung ab

```
filter ns="test.foo" | sort millis desc | limit 10
```

Ruft alle Aktualisierungsvorgänge für eine Sammlung ab, die mehr als 60 ms gedauert haben

```
filter millis > 60 and op = "update"
```

Holen Sie sich die 10 langsamsten Operationen des letzten Monats

```
sort millis desc | limit 10
```

Abrufen aller Abfragen mit einer COLLSCAN-Planübersicht

```
filter planSummary="COLLSCAN"
```

Überwachung mit Performance Insights

Performance Insights erweitert die bestehenden Überwachungsfunktionen von Amazon DocumentDB, um die Leistung Ihres Clusters zu veranschaulichen und Ihnen bei der Analyse aller Probleme zu helfen, die sich darauf auswirken. Mit dem Performance Insights-Dashboard können Sie die Datenbanklast visualisieren und die Auslastung nach Wartezeiten, Abfrageanweisungen, Hosts oder Anwendung filtern.

> **Note**
>
> Performance Insights ist nur für Amazon DocumentDB 3.6-, 4.0-, 5.0- und 8.0-Instance-Cluster verfügbar.

Wie ist es nützlich?

- Datenbankleistung visualisieren — Visualisieren Sie die Last, um festzustellen, wann und wo sich die Last auf der Datenbank befindet
- Ermitteln Sie, was die Datenbank belastet — Ermitteln Sie, welche Abfragen, Hosts und Anwendungen zur Belastung Ihrer Instance beitragen
- Stellen Sie fest, wann Ihre Datenbank belastet ist — Vergrößern Sie das Performance Insights-Dashboard, um sich auf bestimmte Ereignisse zu konzentrieren, oder verkleinern Sie die Ansicht, um Trends über einen längeren Zeitraum zu betrachten
- Warnung bei Datenbanklast — Greifen Sie automatisch auf neue Datenbanklast-Metriken zu, von CloudWatch wo aus Sie die DB-Lademetriken zusammen mit anderen Amazon DocumentDB-Metriken überwachen und entsprechende Warnmeldungen einrichten können

Was sind die Einschränkungen von Amazon DocumentDB Performance Insights?

- Performance Insights in den Regionen AWS GovCloud (US-East) und AWS GovCloud (US-West) sind nicht verfügbar
- Performance Insights for Amazon DocumentDB speichert Leistungsdaten für bis zu 7 Tage
- Abfragen, die länger als 1.024 Byte sind, werden in Performance Insights nicht aggregiert

Themen

- Konzepte von Performance Insights
- Aktivieren und Deaktivieren von Performance Insights
- Konfigurieren von Zugriffsrichtlinien für Performance Insights
- Analyse der Metriken mit dem Performance Insights-Dashboard
- Abrufen von Metriken mit der Performance-Insights-API
- CloudWatch Amazon-Metriken für Performance Insights
- Leistungseinblicke für Zählerkennzahlen

Konzepte von Performance Insights

Themen

- Durchschnittliche aktive Sitzungen
- Dimensionen
- Maximale vCPU

Durchschnittliche aktive Sitzungen

Datenbanklast (DB-Last) misst den Aktivitätsgrad in Ihrer Datenbank. Die wichtigste Metrik für Performance Insights ist `DB Load`, die jede Sekunde erfasst wird. Die Einheit für die `DBLoad` Metrik ist der Durchschnitt der aktiven Sitzungen (AAS) für eine Amazon DocumentDB-Instance.

Eine aktive Sitzung ist eine Verbindung, die Arbeit an die Amazon DocumentDB-Instance übermittelt hat und auf eine Antwort wartet. Wenn Sie beispielsweise eine Anfrage an eine Amazon DocumentDB-Instance senden, ist die Datenbanksitzung aktiv, während die Instance die Abfrage verarbeitet.

Um die Anzahl der durchschnittlich aktiven Sitzungen AAS zu erhalten, ruft Performance Insights die Anzahl der Sitzungen ab, die gleichzeitig eine Abfrage ausführen. Die AAS ist die Gesamtzahl der Sitzungen geteilt durch die Gesamtzahl der Beispiele. Die folgende Tabelle zeigt fünf aufeinanderfolgende Beispiele einer laufenden Abfrage.

Beispiel	Anzahl der Sitzungen, die eine Abfrage ausführen	AAS	Berechnung
1	2	2	2 Sitzungen/1 Beispiel
2	0	1	2 Sitzungen /2 Beispiele
3	4	2	6 Sitzungen /3 Beispiele
4	0	1.5	6 Sitzungen /4 Beispiele

Beispiel	Anzahl der Sitzungen, die eine Abfrage ausführen	AAS	Berechnung
5	4	2	10 Sitzungen /5 Beispiele

Im vorherigen Beispiel beträgt die DB-Auslastung für das Zeitintervall von 1-5 2 AAS. Eine Erhöhung der DB-Last bedeutet, dass im Durchschnitt mehr Sitzungen für die Datenbank ausgeführt werden.

Dimensionen

Die`DB Load` Metrik unterscheidet sich von den anderen Zeitreihenmetriken, da Sie sie in Unterkomponenten aufteilen können, die als Dimensionen bezeichnet werden. Sie können sich Dimensionen als Kategorien für die verschiedenen Merkmale der `DB Load`-Metrik vorstellen. Bei der Diagnose von Leistungsproblemen sind die Dimensionen Wait States und Top Query am nützlichsten.

Wartezustände

Ein Wartezustand bewirkt, dass eine Abfrageanweisung auf das Eintreten eines bestimmten Ereignisses wartet, bevor sie weiter ausgeführt werden kann. Beispielsweise kann eine Abfrageanweisung warten, bis eine gesperrte Ressource entsperrt wird. Durch Kombination `DB Load` mit Wartezuständen können Sie sich ein vollständiges Bild des Sitzungsstatus machen. Hier sind verschiedene Amazon DocumentDB-Wartestatus:

Amazon DocumentDB-Wartestatus	Beschreibung des Wartestatus
Verriegeln	Der Latch-Wartestatus tritt ein, wenn die Sitzung darauf wartet, den Pufferpool zu durchsuchen. Häufiges Ein- und Auslagern des Pufferpools kann häufiger vorkommen, wenn das System häufig umfangreiche Abfragen verarbeitet, Sammelscans durchführt oder wenn der Pufferpool zu klein ist, um den Arbeitssatz zu verarbeiten.

Amazon DocumentDB-Wartestatus	Beschreibung des Wartestatus
CPU	Der CPU-Wartestatus tritt ein, wenn die Sitzung auf die CPU wartet.
CollectionLock	Der CollectionLock Wartezustand tritt ein, wenn die Sitzung darauf wartet, eine Sperre für die Sammlung zu erhalten. Diese Ereignisse treten auf, wenn DDL-Operationen für die Sammlung ausgeführt werden.
DocumentLock	Der DocumentLock Wartestatus tritt ein, wenn die Sitzung darauf wartet, eine Sperre für ein Dokument zu erhalten. Eine hohe Anzahl gleichzeitiger Schreibvorgänge in dasselbe Dokument führt zu mehr DocumentLock Wartezuständen für dieses Dokument.
SystemLock	Der SystemLock Wartestatus tritt ein, wenn die Sitzung auf dem System wartet. Dies kann auftreten, wenn häufig Abfragen mit langer Laufzeit, Transaktionen mit langer Laufzeit oder hoher Parallelität auf dem System ausgeführt werden.
IO	Der IO-Wartestatus tritt ein, wenn die Sitzung darauf wartet, dass IO abgeschlossen ist.
BufferLock	Der BufferLock Wartezustand tritt ein, wenn die Sitzung darauf wartet, eine gemeinsam genutzte Seite im Puffer zu sperren. BufferLoc kDer Wartestatus kann verlängert werden, wenn andere Prozesse die Cursor auf den angeforderten Seiten geöffnet halten.

Amazon DocumentDB-Wartestatus	Beschreibung des Wartestatus
LowMemThrottle	Der LowMemThrottle Wartezustand tritt ein, wenn die Sitzung aufgrund einer starken Speicherauslastung der Amazon DocumentDB-Instance wartet. Wenn dieser Zustand über einen längeren Zeitraum anhält, sollten Sie erwägen, die Instance zu skalieren, um zusätzlichen Speicher bereitzustellen. Weitere Informationen finden Sie unter Resource Governor.
BackgroundActivity	Der BackgroundActivity Wartezustand tritt ein, wenn die Sitzung auf interne Systemprozesse wartet.
Sonstige	Der Wartestatus Anderer ist ein interner Wartezustand. Wenn dieser Zustand lange anhält, sollten Sie erwägen, diese Abfrage zu beenden. Weitere Informationen finden Sie unter Wie finde und beende ich lang andauernde oder blockierte Abfragen?

Die häufigsten Suchanfragen

Während Wartestatus Engpässe anzeigen, zeigen Top-Abfragen, welche Abfragen am meisten zur DB-Auslastung beitragen. Beispielsweise könnten derzeit viele Abfragen gleichzeitig in der Datenbank ausgeführt werden, aber eine einzelne Abfrage könnte 99 % der DB-Last verbrauchen. In diesem Fall könnte die hohe Belastung auf ein Problem mit der Abfrage hinweisen.

Maximale vCPU

Das Diagramm Datenbanklast im Dashboard dient zum Erfassen, Aggregieren und Anzeigen von Sitzungsinformationen. Um zu sehen, ob aktive Sitzungen die maximale CPU überschreiten, sehen Sie sich ihre Beziehung zur Max vCPU-Linie an. Der maximale vCPU-Wert wird durch die Anzahl der vCPU-Kerne (virtuelle CPU) für Ihre Amazon DocumentDB-Instance bestimmt.

Wenn die DB-Last häufig über der Max vCPU-Linie liegt und der primäre Wartezustand „CPU“ lautet, ist die CPU überlastet. In diesem Fall möchten Sie möglicherweise die Verbindungen zur Instance drosseln, alle Abfragen mit einer hohen CPU-Last optimieren oder eine größere Instance-Klasse in Betracht ziehen. Hohe und konsistente Instances von Wartezuständen deuten darauf hin, dass es möglicherweise Engpässe oder Probleme mit Ressourcenkonflikten gibt, die behoben werden müssen. Dies kann auch dann zutreffen, wenn die DB-Last die mit Max vCPU definierte Linie nicht überschreitet.

Aktivieren und Deaktivieren von Performance Insights

Um Performance Insights zu nutzen, aktivieren Sie es auf Ihrer DB-Instance. Sie können es später bei Bedarf deaktivieren. Das Aktivieren und Deaktivieren von Performance Insights führt nicht zu Ausfallzeiten, einem Neustart oder einem Failover.

Der Performance-Insights-Agent verbraucht eine begrenzte Menge an CPU und Arbeitsspeicher auf dem DB-Host. Wenn die DB-Last hoch ist, begrenzt der Agent die Auswirkungen auf die Leistung, indem Daten seltener erfasst werden.

Aktivieren von Performance Insights beim Erstellen eines Clusters

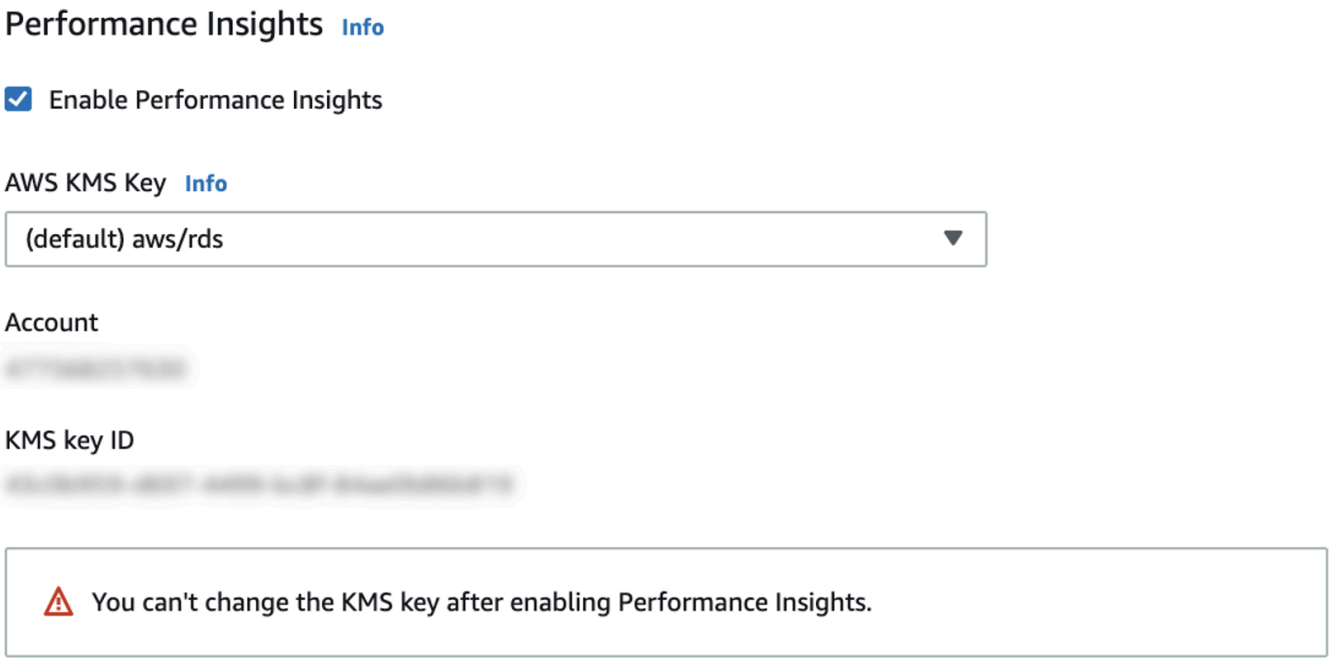
In der Konsole können Sie Performance Insights aktivieren oder deaktivieren, wenn Sie eine neue DB-Instance erstellen oder ändern.

Verwendung der AWS-Managementkonsole

In der Konsole können Sie Performance Insights aktivieren, wenn Sie einen Amazon DocumentDB-Cluster erstellen. Wenn Sie einen neuen Amazon DocumentDB-Cluster erstellen, aktivieren Sie Performance Insights, indem Sie im Abschnitt Performance Insights die Option Enable Performance Insights auswählen.

Anweisungen für die Konsole

1. Um einen Cluster zu erstellen, folgen Sie den Anweisungen zum Erstellen eines Amazon DocumentDB-Clusters.
2. Wählen Sie im Abschnitt Performance Insights die Option Performance Insights aktivieren aus.



> **Note**
>
> Die Aufbewahrungsfrist für Performance Insights-Daten beträgt sieben Tage.

AWS KMS Schlüssel — Geben Sie Ihren AWS KMS-Schlüssel an. Performance Insights verschlüsselt alle potenziell sensiblen Daten mithilfe Ihres AWS KMS Schlüssels. Die Daten werden während der Übertragung und im Ruhezustand verschlüsselt. Weitere Informationen finden Sie unter Konfiguration einer AWS AWS KMS Richtlinie für Performance Insights.

Aktivierung und Deaktivierung beim Ändern einer Instanz

Sie können eine DB-Instance ändern, um Performance Insights mithilfe der Konsole oder zu aktivieren oder AWS CLI zu deaktivieren.

Using the AWS-Managementkonsole

Anweisungen für die Konsole

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Klicken Sie auf Cluster.

3. Wählen Sie eine DB-Instance aus und wählen Sie Ändern.
4. Wählen Sie im Abschnitt Performance Insights entweder Performance Insights aktivieren oder Performance Insights deaktivieren aus.

   > **Note**
   >
   > Wenn Sie Performance Insights aktivieren wählen, können Sie Ihren AWS AWS KMS Schlüssel angeben. Performance Insights verschlüsselt alle potenziell sensiblen Daten mithilfe Ihres AWS KMS Schlüssels. Die Daten werden während der Übertragung und im Ruhezustand verschlüsselt. Weitere Informationen finden Sie unter Verschlüsseln von Amazon DocumentDB-Daten im Ruhezustand.

5. Klicken Sie auf Weiter.
6. Wählen Sie für Einplanung von Änderungen die Option Sofort anwenden aus. Wenn Sie im nächsten geplanten Wartungsfenster Anwenden wählen, ignoriert Ihre Instance diese Einstellung und aktiviert Performance Insights sofort.
7. Wählen Sie Modify instance (Instance ändern).

Using the AWS CLI

Wenn Sie die `modify-db-instance` AWS AWS CLI Befehle `create-db-instance` oder verwenden, können Sie Performance Insights aktivieren, indem Sie sie angeben`--enable-performance-insights`, oder deaktivieren, indem Sie Folgendes angeben`--no-enable-performance-insights`.

Im folgenden Verfahren wird beschrieben, wie Sie mit der AWS AWS CLI Performance-Insights für eine DB-Instance aktivieren oder deaktivieren.

AWS AWS CLI Anweisungen

Rufen Sie den `modify-db-instance` AWS AWS CLI Befehl auf und geben Sie die folgenden Werte ein:

- `--db-instance-identifer`— Der Name der DB-Instance
- `--enable-performance-insights` zum Aktivieren oder `--no-enable-performance-insights` zum Deaktivieren

Example

Das folgende Beispiel aktiviert Performance Insights für`sample-db-instance`:

For Linux, macOS, or Unix:

```
aws docdb modify-db-instance \
    --db-instance-identifier sample-db-instance \
    --enable-performance-insights
```

For Windows:

```
aws docdb modify-db-instance ^
    --db-instance-identifier sample-db-instance ^
    --enable-performance-insights
```

Konfigurieren von Zugriffsrichtlinien für Performance Insights

Um auf Performance Insights zugreifen zu können, müssen Sie über die entsprechenden Berechtigungen von AWS Identity and Access Management (IAM) verfügen. Sie haben folgende Möglichkeiten, Zugriff zu gewähren:

- Fügen Sie die verwaltete Richtlinie `AmazonRDSPerformanceInsightsReadOnly` an einen Berechtigungssatz oder eine Rolle an.
- Erstellen Sie eine benutzerdefinierte IAM-Richtlinie und fügen Sie diese an einen Berechtigungssatz oder eine Rolle an.

Wenn Sie bei der Aktivierung von Performance Insights einen vom Kunden verwalteten Schlüssel angegeben haben, stellen Sie außerdem sicher, dass die Benutzer in Ihrem Konto über die Berechtigungen `kms:Decrypt` und `kms:GenerateDataKey` für den KMS-Schlüssel verfügen.

> **Note**
>
> Für die Verschlüsselung im Ruhezustand mit AWS KMS Schlüsseln und Sicherheitsgruppen nutzt Amazon DocumentDB betriebliche Technologien, die mit Amazon RDS gemeinsam genutzt werden. https://aws.amazon.com/rds

Die Richtlinie an einen IAM-Prinzipal anhängen AmazonRDSPerformanceInsightsReadOnly

`AmazonRDSPerformanceInsightsReadOnly`ist eine AWS verwaltete Richtlinie, die Zugriff auf alle schreibgeschützten Operationen der Amazon DocumentDB Performance Insights-API gewährt. Derzeit sind alle Operationen in dieser API schreibgeschützt. Wenn Sie `AmazonRDSPerformanceInsightsReadOnly` an einen Berechtigungssatz oder eine Rolle anfügen, kann der Empfänger Performance Insights mit anderen Konsolenfunktionen verwenden.

Erstellen einer benutzerdefinierten IAM-Richtlinie für Performance Insights

Für Benutzer, die nicht über die `AmazonRDSPerformanceInsightsReadOnly`-Richtlinien verfügen, können Sie den Zugriff auf Performance Insights gewähren, indem Sie eine benutzerverwaltete IAM-Richtlinie erstellen oder ändern. Wenn Sie die Richtlinie an einen Berechtigungssatz oder eine Rolle anhängen, kann der Empfänger Performance Insights verwenden.

Erstellen eine benutzerdefinierten Richtlinie

1. Öffnen Sie unter https://console.aws.amazon.com/iam/ die IAM-Konsole.
2. Wählen Sie im Navigationsbereich Richtlinien.
3. Wählen Sie Richtlinie erstellen aus.
4. Wählen Sie auf der Seite Create Policy (Richtlinie erstellen) die Registerkarte „JSON“ aus.
5. Kopieren Sie den folgenden Text und fügen Sie ihn ein. *`us-east-1`* Ersetzen Sie ihn durch den Namen Ihrer AWS Region und *`111122223333`* durch Ihre Kundenkontonummer.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "rds:DescribeDBInstances",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
```

```
            "Action": "rds:DescribeDBClusters",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "pi:DescribeDimensionKeys",
            "Resource": "arn:aws:pi:us-east-1:111122223333:metrics/rds/*"
        },
        {
            "Effect": "Allow",
            "Action": "pi:GetDimensionKeyDetails",
            "Resource": "arn:aws:pi:us-east-1:111122223333:metrics/rds/*"
        },
        {
            "Effect": "Allow",
            "Action": "pi:GetResourceMetadata",
            "Resource": "arn:aws:pi:us-east-1:111122223333:metrics/rds/*"
        },
        {
            "Effect": "Allow",
            "Action": "pi:GetResourceMetrics",
            "Resource": "arn:aws:pi:us-east-1:111122223333:metrics/rds/*"
        },
        {
            "Effect": "Allow",
            "Action": "pi:ListAvailableResourceDimensions",
            "Resource": "arn:aws:pi:us-east-1:111122223333:metrics/rds/*"
        },
        {
            "Effect": "Allow",
            "Action": "pi:ListAvailableResourceMetrics",
            "Resource": "arn:aws:pi:us-east-1:111122223333:metrics/rds/*"
        }
    ]
}
```

6. Wählen Sie Richtlinie prüfen.
7. Geben Sie einen Namen und optional eine Beschreibung für die Richtlinie an und wählen Sie dann Create policy (Richtlinie erstellen) aus.

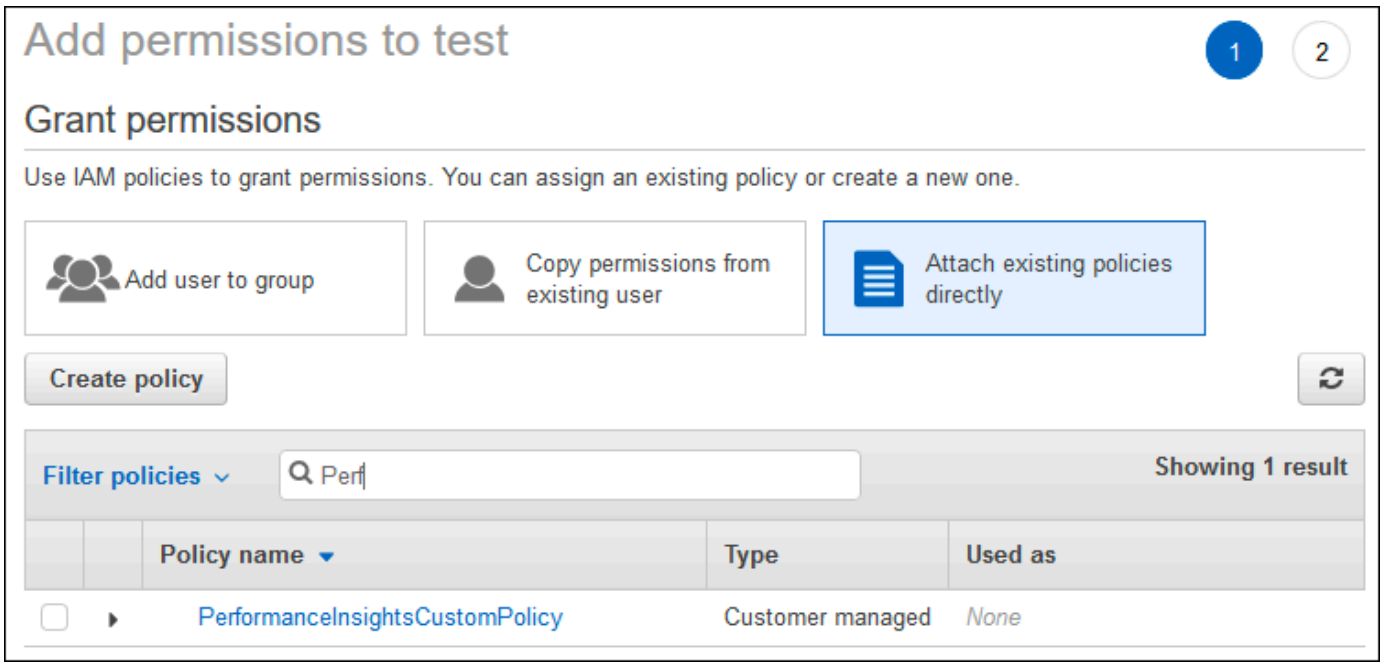
Sie können die Richtlinie nun an einen Berechtigungssatz oder eine Rolle anfügen. Das folgende Verfahren setzt voraus, dass Sie für diesen Zweck bereits einen Benutzer zur Verfügung haben.

So fügen Sie die Richtlinie an einen Benutzer an

1. Öffnen Sie unter https://console.aws.amazon.com/iam/ die IAM-Konsole.
2. Klicken Sie im Navigationsbereich auf Users (Benutzer).
3. Wählen Sie einen vorhandenen Benutzer aus der Liste aus.

> **Important**
>
> Um Performance Insights verwenden zu können, stellen Sie sicher, dass Sie zusätzlich zur benutzerdefinierten Richtlinie Zugriff auf Amazon DocumentDB haben. Die AmazonDocDBReadOnlyAccess vordefinierte Richtlinie bietet beispielsweise nur Lesezugriff auf Amazon. DocDB.For Weitere Informationen finden Sie unter Zugriff mithilfe von Richtlinien verwalten.

4. Wählen Sie auf der Seite Summary (Übersicht) die Option Add permissions (Berechtigungen hinzufügen) aus.
5. Wählen Sie Attach existing policies directly (Vorhandene Richtlinien direkt zuordnen). Geben Sie in Suche die ersten Zeichen Ihres Richtliniennamens ein, wie nachfolgend gezeigt.



6. Wählen Sie Ihre Richtlinie und wählen Sie anschließend Nächster Schritt: Prüfen.
7. Wählen Sie Add permissions (Berechtigungen hinzufügen) aus.

Konfiguration einer AWS KMS Richtlinie für Performance Insights

Performance Insights verwendet ein AWS KMS key , um sensible Daten zu verschlüsseln. Wenn Sie Performance Insights über die API oder die Konsole aktivieren, haben Sie folgende Möglichkeiten:

- Wählen Sie die Standardeinstellung Von AWS verwalteter Schlüssel.

  Amazon DocumentDB verwendet das Von AWS verwalteter Schlüssel für Ihre neue DB-Instance. Amazon DocumentDB erstellt eine Von AWS verwalteter Schlüssel für Ihr AWS Konto. Ihr AWS Konto hat Von AWS verwalteter Schlüssel für jede AWS Region ein anderes Amazon DocumentDB-Konto.
- Wählen Sie einen kundenverwalteten Schlüssel.

  Wenn Sie einen vom Kunden verwalteten Schlüssel angeben, benötigen Benutzer in Ihrem Konto, die die Performance Insights API aufrufen, die Berechtigungen `kms:Decrypt` und `kms:GenerateDataKey` für den KMS-Schlüssel. Verwalten Sie die Berechtigungen des KMS-Schlüssels mithilfe der AWS KMS Schlüsselrichtlinie statt mithilfe der IAM-Richtlinien. Weitere Informationen finden Sie unter Verwenden wichtiger Richtlinien in AWS KMS.

Example

Das folgende Beispiel für eine Schlüsselrichtlinie zeigt, wie Sie Ihrer KMS-Schlüsselrichtlinie Anweisungen hinzufügen können. Diese Anweisungen erlaubt den Zugriff auf Performance Insights. Je nachdem, wie Sie den verwenden AWS KMS, möchten Sie möglicherweise einige Einschränkungen ändern. Bevor Sie Ihrer Richtlinie Anweisungen hinzufügen, entfernen Sie alle Kommentare.

Analyse der Metriken mit dem Performance Insights-Dashboard

Das Dashboard von Performance Insights enthält Informationen zur Datenbank-Performance, die Sie bei der Analyse und Behebung von Performance-Problemen unterstützen. Auf der Hauptseite des Dashboards können Sie Informationen zur Datenbanklast (DB-Last) einsehen. Sie können die DB-Last nach Dimensionen wie Wartestatus oder Abfrage „aufteilen“.

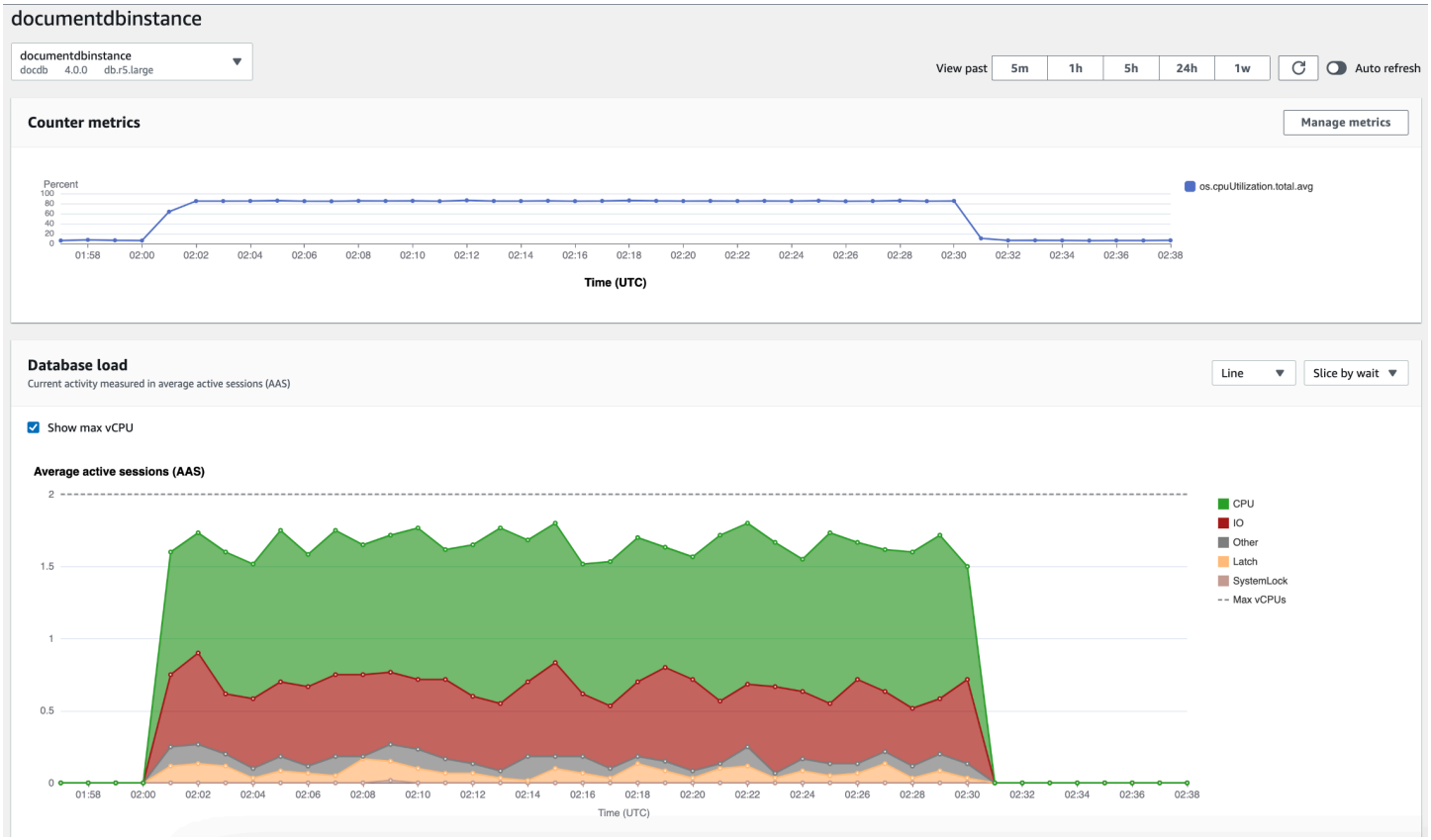
Themen

- Überblick über Performance Insights
- Öffnen des Performance Insights-Dashboards
- Analysiert die Datenbanklast anhand von Wartezuständen

- Überblick über den Tab „Häufigste Abfragen“
- Das Datenbank-Ladediagramm vergrößern

Überblick über Performance Insights

Das Dashboard ist die einfachste Möglichkeit, mit Performance Insights zu interagieren. Das folgende Beispiel zeigt das Dashboard für eine Amazon DocumentDB-Instance. Standardmäßig zeigt das Dashboard von Performance Insights die Daten der letzten Stunde an.



Das Dashboard ist in folgende Teile gegliedert:

1. Zählermesswerte — Zeigt Daten für bestimmte Leistungsindikator-Metriken an.
2. Datenbanklast — Zeigt, wie die DB-Auslastung mit der DB-Instance-Kapazität verglichen wird, die durch die Zeile Max. vCPU dargestellt wird.
3. Oberste Dimensionen — Zeigt die obersten Dimensionen an, die zur DB-Auslastung beitragen. Zu diesen Dimensionen gehören `waits` `queries`,`hosts`,`databases`, und`applications`.

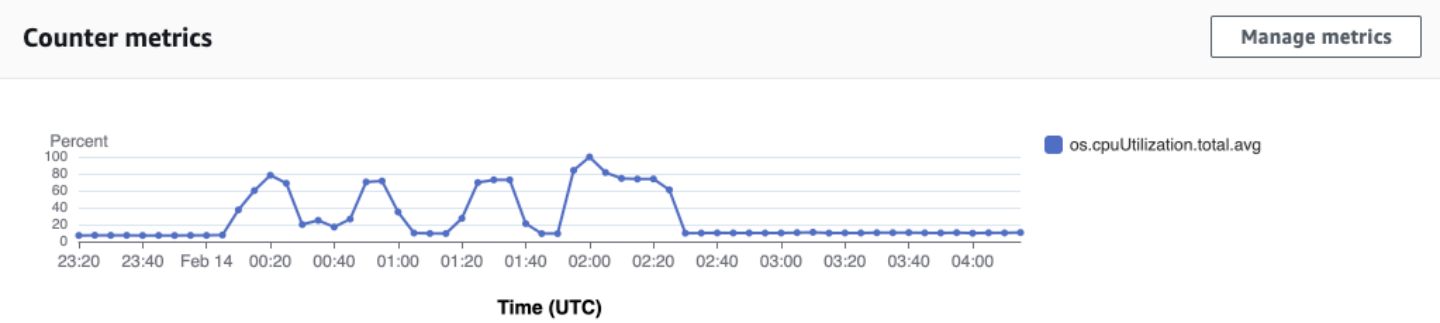
Themen

- Zählermetriken-Diagramm
- Datenbank-Ladediagramm
- Dimensionen pro Tabelle

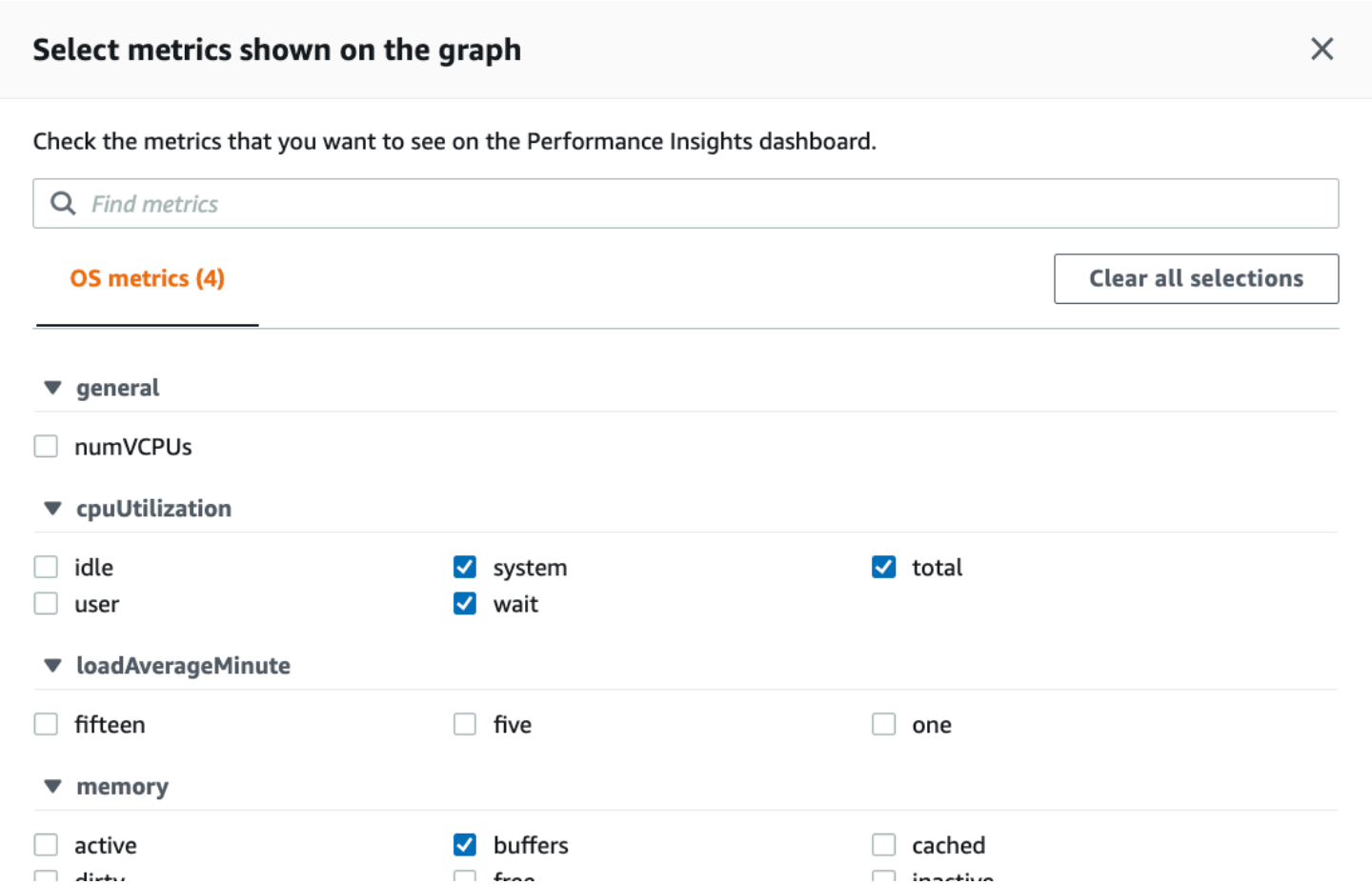
Zählermetriken-Diagramm

Mithilfe von Zählermetriken können Sie das Performance-Insights-Dashboard anpassen und bis zu 10 weitere Diagramme aufnehmen. Diese Grafiken zeigen eine Auswahl von Dutzenden von Betriebssystemmetriken. Diese Informationen können mit der Datenbanklast korreliert werden, um Performance-Probleme zu identifizieren und zu analysieren.

Das Counter Metrics (Zählermetriken)-Diagramm enthält Daten zu Leistungsindikatoren.



Um die Leistungsindikatoren zu ändern, wählen Sie Messwerte verwalten aus. Sie können mehrere Betriebssystem-Metriken auswählen, wie im folgenden Screenshot gezeigt. Um Details für jede Metrik anzuzeigen, bewegen Sie den Mauszeiger über den Metriknamen.



Datenbank-Ladediagramm

Das Datenbanklastdiagramm zeigt, wie die Datenbankaktivität im Vergleich zur Instanzkapazität verglichen wird, die durch die Zeile Max. vCPU dargestellt wird. Standardmäßig stellt das gestapelte Liniendiagramm die DB-Last als durchschnittliche aktive Sitzungen pro Zeiteinheit dar. Die DB-Last wird nach Wartestatus aufgeteilt (gruppiert).



DB-Last aufgeteilt nach Dimensionen

Sie können die Last als aktive Sitzungen anzeigen, die nach unterstützten Dimensionen gruppiert sind. Die folgende Abbildung zeigt die Abmessungen der Amazon DocumentDB-Instance.

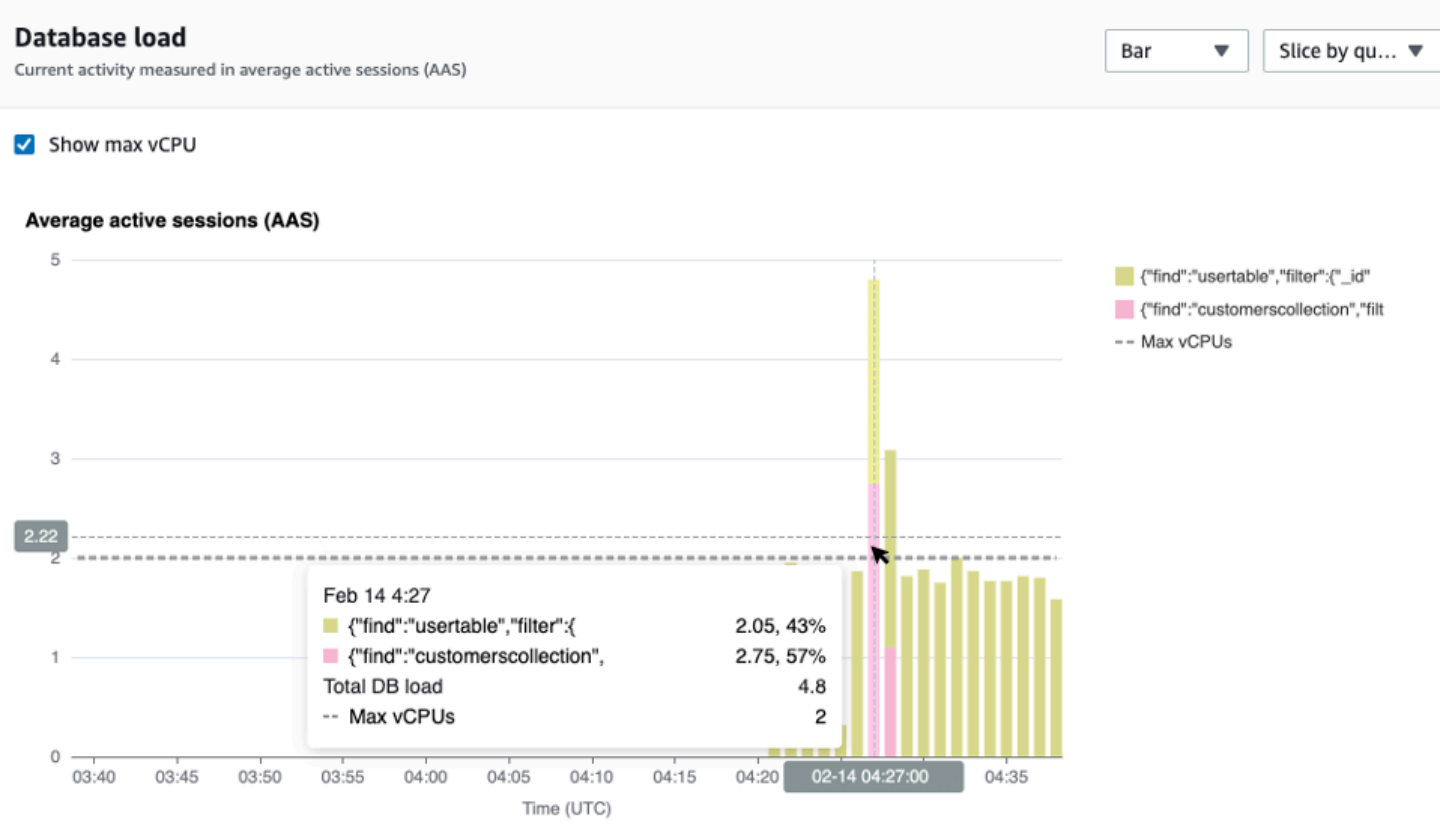


DB-Ladedetails für ein Dimensionselement

Um Details zu einem DB-Lastelement innerhalb einer Dimension anzuzeigen, bewegen Sie den Mauszeiger über den Elementnamen. Die folgende Abbildung zeigt Details für eine Abfrageanweisung.



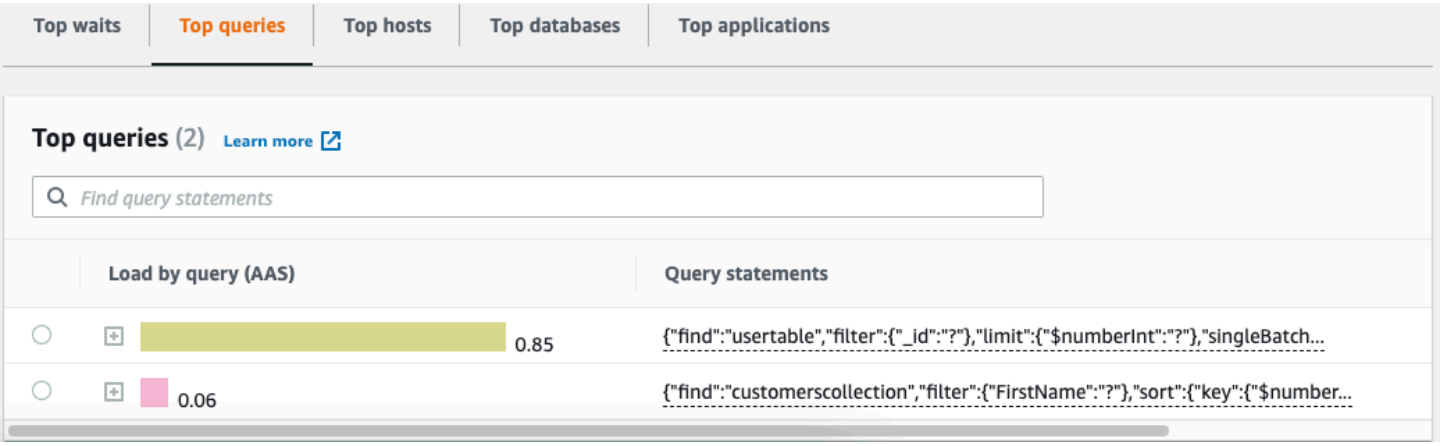
Um Details zu einem Element für den ausgewählten Zeitraum in der Legende anzuzeigen, bewegen Sie den Mauszeiger über dieses Element.



Dimensionen pro Tabelle

In der Tabelle mit den obersten Dimensionen wird die DB-Last nach verschiedenen Dimensionen aufgeteilt. Eine Dimension ist eine Kategorie oder „Aufteilung“ für verschiedene Merkmale der DB-Last. Wenn es sich bei der Dimension um eine Abfrage handelt, werden in den Top-Abfragen die Abfrageanweisungen angezeigt, die am meisten zur DB-Auslastung beitragen.

Wählen Sie eine der folgenden Dimensionsregisterkarten.



Die folgende Tabelle enthält eine kurze Beschreibung der einzelnen Registerkarten.

Dimension	Description
Das wartende Ereignis, auf das die Datenbank-Backend wartet	Das wartende Ereignis (Top-Wait-Ereignis)
Die Anweisungen, Abfragen die derzeit ausgeführt werden	Die Abfragen (Top-SQL)
Die Hosts (IP-Adressen der verbundenen Clients	Die Hosts (Top-Hosts)
Der Name der Datenbank	Die Datenbanken (Top-Datenbanken)

Dat	Beschreibung
Dat	Datenbanken, mit der der Client verbunden ist
Top Anwendungen (Top-Anwendungen)	Der Name der Anwendung, die mit der Datenbank verbunden ist

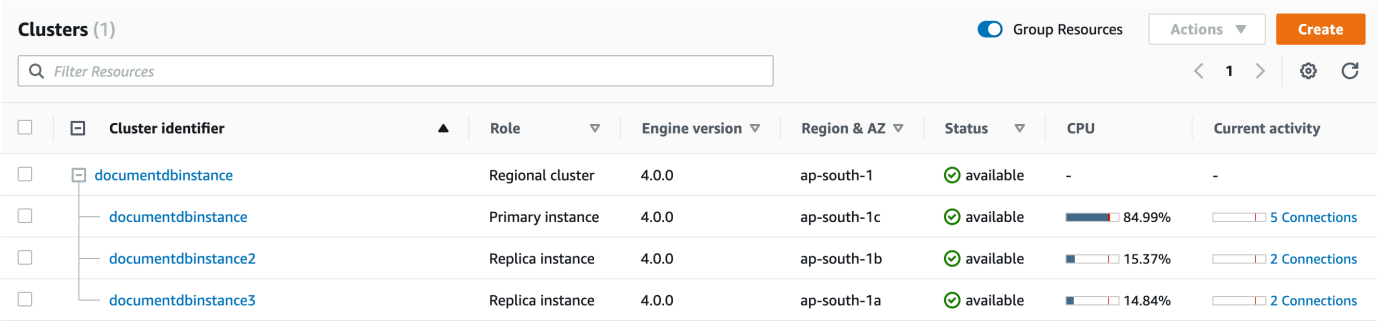
Informationen zum Analysieren von Abfragen mithilfe der Registerkarte „Häufigste Abfragen“ finden Sie unter Überblick über den Tab „Häufigste Abfragen“.

Öffnen des Performance Insights-Dashboards

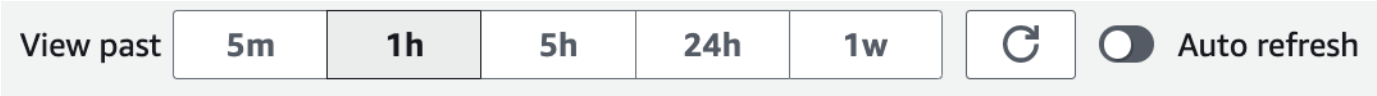
Um das Performance Insights-Dashboard im AWS Gehen Sie in der Managementkonsole wie folgt vor:

1. Öffnen Sie die Performance Insights-Konsole unter https://console.aws.amazon.com/docdb/.
2. Wählen Sie eine DB-Instance aus. Das Performance Insights-Dashboard wird für diese Amazon DocumentDB-Instance angezeigt.

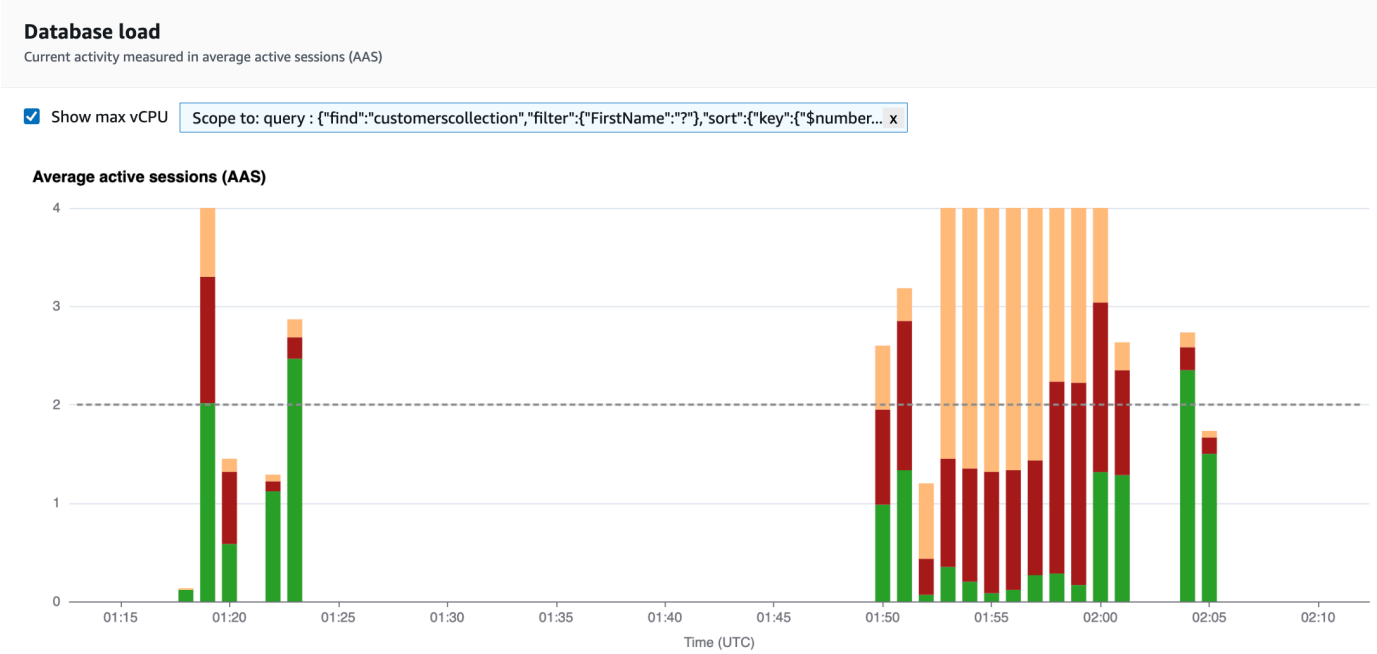
Für Amazon DocumentDB-Instances mit aktiviertem Performance Insights können Sie das Dashboard auch erreichen, indem Sie das Element Sessions in der Liste der Instances auswählen. Unter Aktuelle Aktivität zeigt das Element Sitzungen die Datenbanklast von durchschnittlichen, aktiven Sitzungen der letzten fünf Minuten an. Der Balken zeigt die Last grafisch an. Wenn die Leiste leer ist, ist die Instance inaktiv. Wenn die Last ansteigt, wird der Balken blau ausgefüllt. Wenn die Last die Anzahl der virtuellen CPUs (vCPUs) in der Instanzklasse überschreitet, färbt sich der Balken rot, was auf einen potenziellen Engpass hinweist.



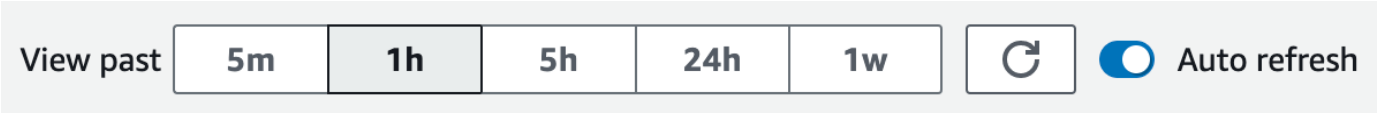
3. (Optional) Wählen Sie über eine der Schaltflächen rechts oben ein anderes Zeitintervall aus. Um das Intervall beispielsweise auf 1 Stunde zu ändern, wählen Sie 1h aus.



Im folgenden Screenshot beträgt das DB-Ladeintervall 1 Stunde.



4. Um Ihre Daten automatisch zu aktualisieren, aktivieren Sie Automatische Aktualisierung.



   Das Performance Insights-Dashboard wird automatisch mit neuen Daten aktualisiert. Die Aktualisierungsrate hängt von der Menge der angezeigten Daten ab:

   - „5 Minuten“ wird alle 5 Sekunden aktualisiert.
   - 1 Stunde, wird jede Minute aktualisiert.
   - 5 Stunden, wird jede Minute aktualisiert.
   - „24 Stunden“ wird alle 5 Minuten aktualisiert.
   - „1 Woche“ wird jede Stunden aktualisiert.

Analysiert die Datenbanklast anhand von Wartezuständen

Wenn das Diagramm zur Datenbanklast (DB-Auslastung) einen Engpass zeigt, können Sie herausfinden, woher die Last kommt. Betrachten Sie dazu die Tabelle mit den Hauptlastelementen unterhalb des Datenbanklast-Diagramms Wählen Sie ein bestimmtes Element aus, z. B. eine Abfrage oder eine Anwendung, um dieses Element genauer zu untersuchen und Details dazu anzuzeigen.

Die DB-Auslastung, gruppiert nach Wartezeiten und häufigsten Abfragen, bietet in der Regel den besten Einblick in Leistungsprobleme. DB-Last gruppiert nach Wartezeiten zeigt an, ob Ressourcen- oder Parallelitätsengpässe in der Datenbank vorhanden sind. In diesem Fall wird auf der Registerkarte „Häufigste Abfragen“ der Tabelle mit den am häufigsten ausgelasteten Elementen angezeigt, welche Abfragen für die Auslastung verantwortlich sind.

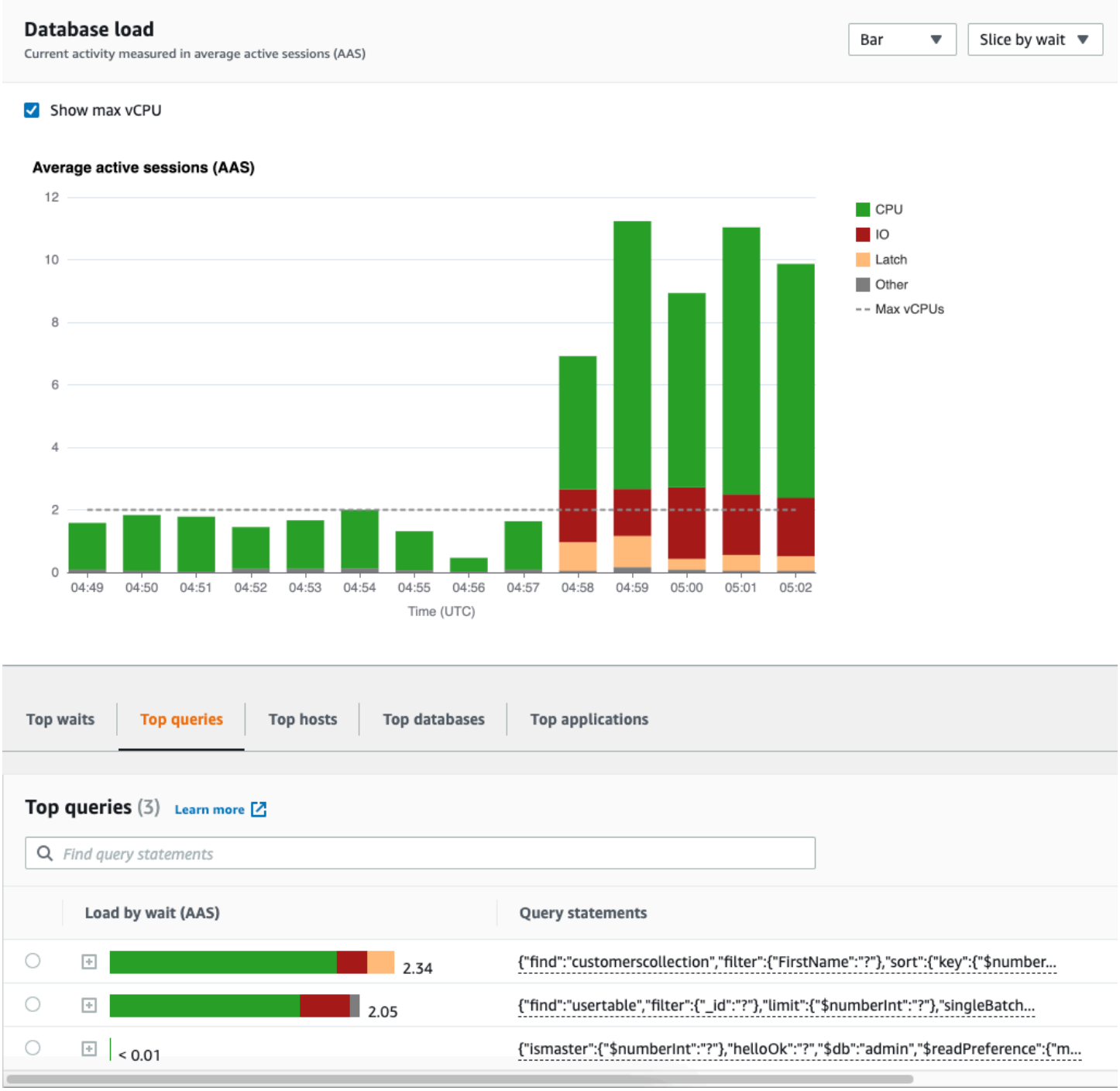
Ihr typischer Workflow für die Diagnose von Performance-Problemen ist folgendermaßen:

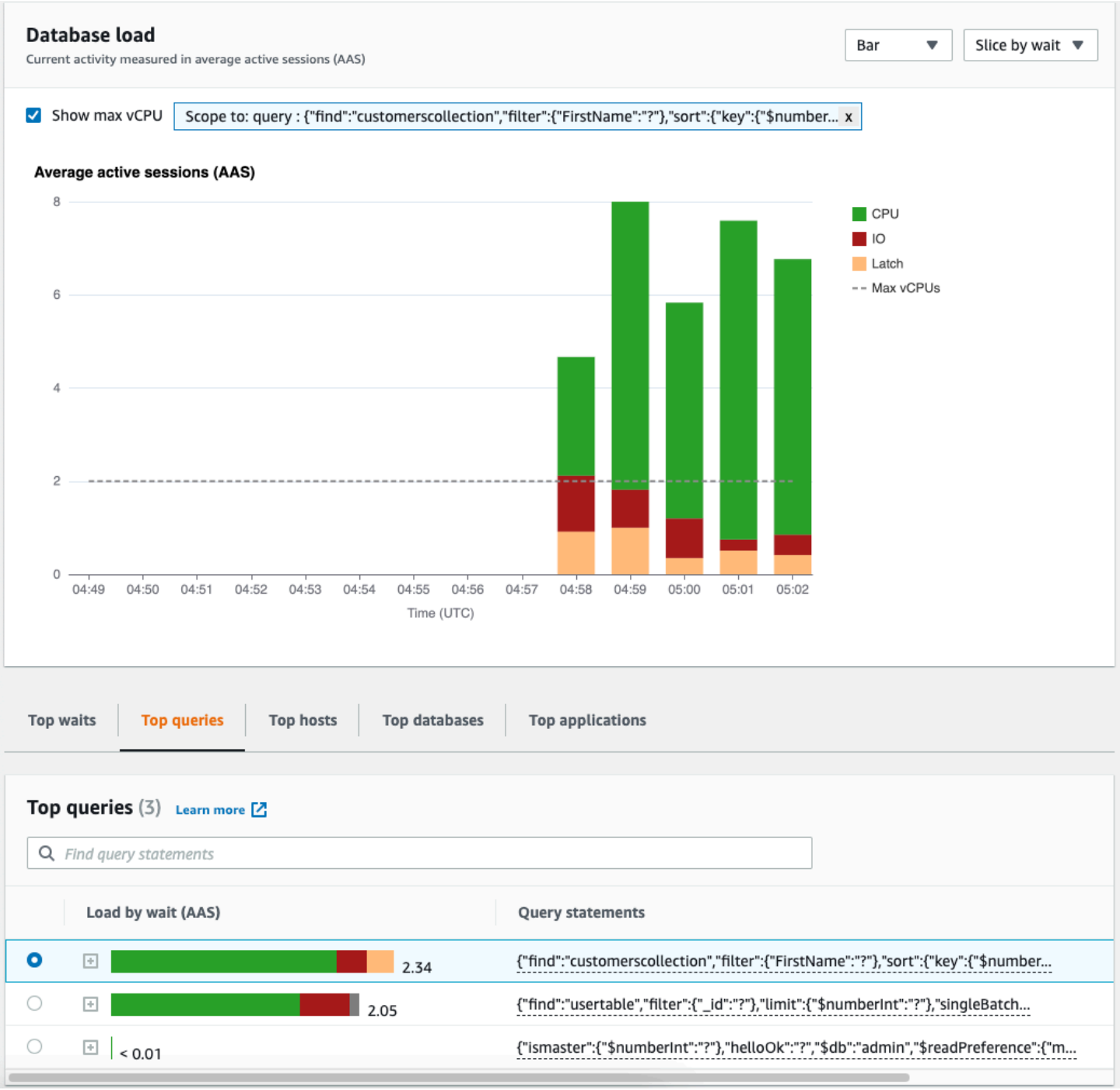
1. Überprüfen Sie das Diagramm der durchschnittlich aktiven Sitzungen auf irgendwelche Ereignisse, in denen die Datenbanklast die Max CPU-Linie übersteigt.
2. Wenn ja, schauen Sie sich das Diagramm der durchschnittlich aktiven Sitzungen an und identifizieren Sie, welcher Wartezustand oder welche Zustände primär dafür verantwortlich sind.
3. Identifizieren Sie die Digest-Abfragen, die das Laden verursacht haben, indem Sie sehen, welche der Abfragen auf der Registerkarte „Häufigste Abfragen“ in der Tabelle mit den häufigsten Ladeelementen am meisten zu diesen Wartezuständen beitragen. Sie können diese anhand der Spalte Load by Wait (AAS) identifizieren.

4. Wählen Sie auf der Registerkarte „Häufigste Abfragen“ eine dieser Digest-Abfragen aus, um sie zu erweitern und die untergeordneten Abfragen anzuzeigen, aus denen sie besteht.

Sie können auch sehen, welche Hosts oder Anwendungen am meisten zur Last beitragen, indem Sie Top-Hosts bzw. Top-Anwendungen auswählen. Die Anwendungsnamen werden in der Verbindungszeichenfolge zur Amazon DocumentDB-Instance angegeben. Unknowngibt an, dass das Anwendungsfeld nicht angegeben wurde.

Im folgenden Dashboard machen beispielsweise CPU-Wartezeiten den größten Teil der DB-Last aus. Wenn Sie die oberste Abfrage unter Häufigste Abfragen auswählen, wird das Datenbanklastdiagramm so angepasst, dass der Schwerpunkt auf der höchsten Last liegt, die durch die Auswahlabfrage verursacht wird.


Database load
Current activity measured in average active sessions (AAS)
Bar
Slice by wait
Show max vCPU
Average active sessions (AAS)
12
10
8
6
4
2
0
CPU
IO
Latch
Other
Max vCPUs
04:49
04:50
04:51
04:52
04:53
04:54
04:55
04:56
04:57
04:58
04:59
05:00
05:01
05:02
Time (UTC)
Top waits
Top queries
Top hosts
Top databases
Top applications
Top queries (3)
Learn more
Find query statements
Load by wait (AAS)
Query statements
2.34
{"find":"customerscollection","filter":{"FirstName":"?"},"sort":{"key":{"$number...
2.05
{"find":"usertable","filter":{"_id":"?"},"limit":{"$numberInt":"?"},"singleBatch...
< 0.01
{"ismaster":{"$numberInt":"?"},"helloOk":"?","$db":"admin","$readPreference":{"m...



Überblick über den Tab „Häufigste Abfragen“

Standardmäßig werden auf der Registerkarte „Häufigste Abfragen“ die Abfragen angezeigt, die am meisten zur DB-Auslastung beitragen. Sie können den Abfragetext analysieren, um Ihre Abfragen zu optimieren.

Themen

- Zusammenfassungen von Abfragen
- Nach Waits laden (AAS)
- Anzeige detaillierter Abfrageinformationen
- Zugriff auf den Abfragetext der Anweisung
- Abfragetext für eine Aussage anzeigen und herunterladen

Zusammenfassungen von Abfragen

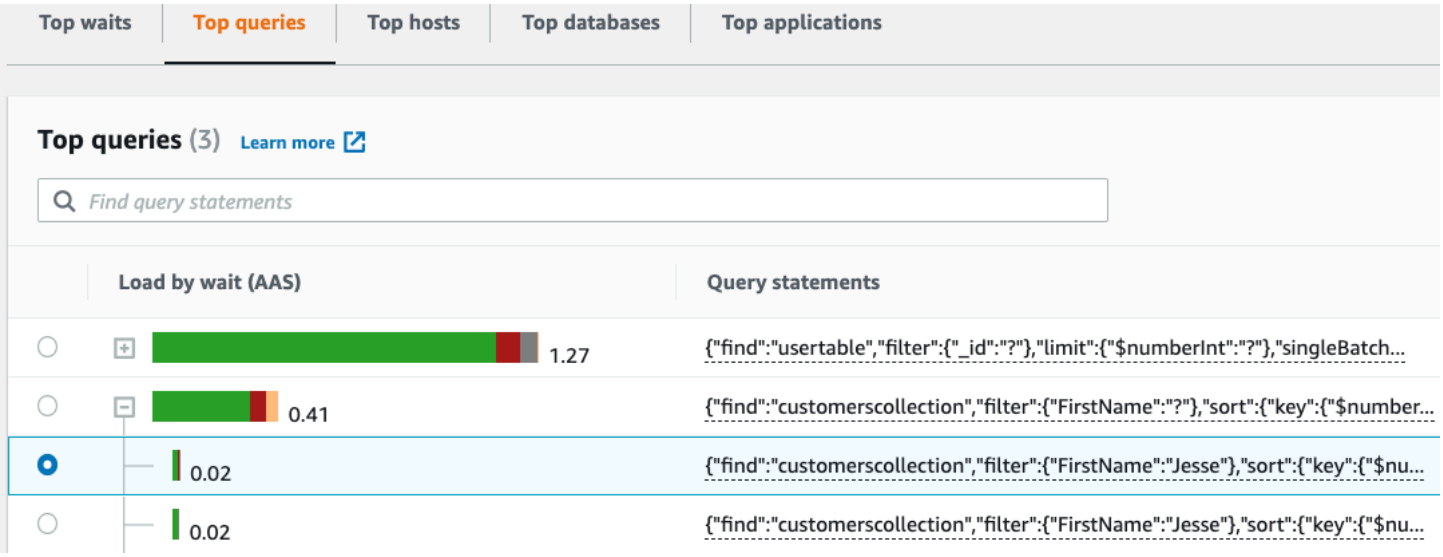
Ein Abfragedigest besteht aus mehreren tatsächlichen Abfragen, die strukturell ähnlich sind, aber unterschiedliche Literalwerte haben können. Der Digest ersetzt fest codierte Werte durch ein Fragezeichen. Ein Abfrage-Digest könnte beispielsweise so aussehen:

```
{"find":"customerscollection","filter":{"FirstName":"?"},"sort":{"key":
{"$numberInt":"?"}},"limit":{"$numberInt":"?"}}
```

Dieser Digest kann die folgenden untergeordneten Abfragen enthalten:

```
{"find":"customerscollection","filter":{"FirstName":"Karrie"},"sort":{"key":
{"$numberInt":"1"}},"limit":{"$numberInt":"3"}}
{"find":"customerscollection","filter":{"FirstName":"Met"},"sort":{"key":
{"$numberInt":"1"}},"limit":{"$numberInt":"3"}}
{"find":"customerscollection","filter":{"FirstName":"Rashin"},"sort":{"key":
{"$numberInt":"1"}},"limit":{"$numberInt":"3"}}
```

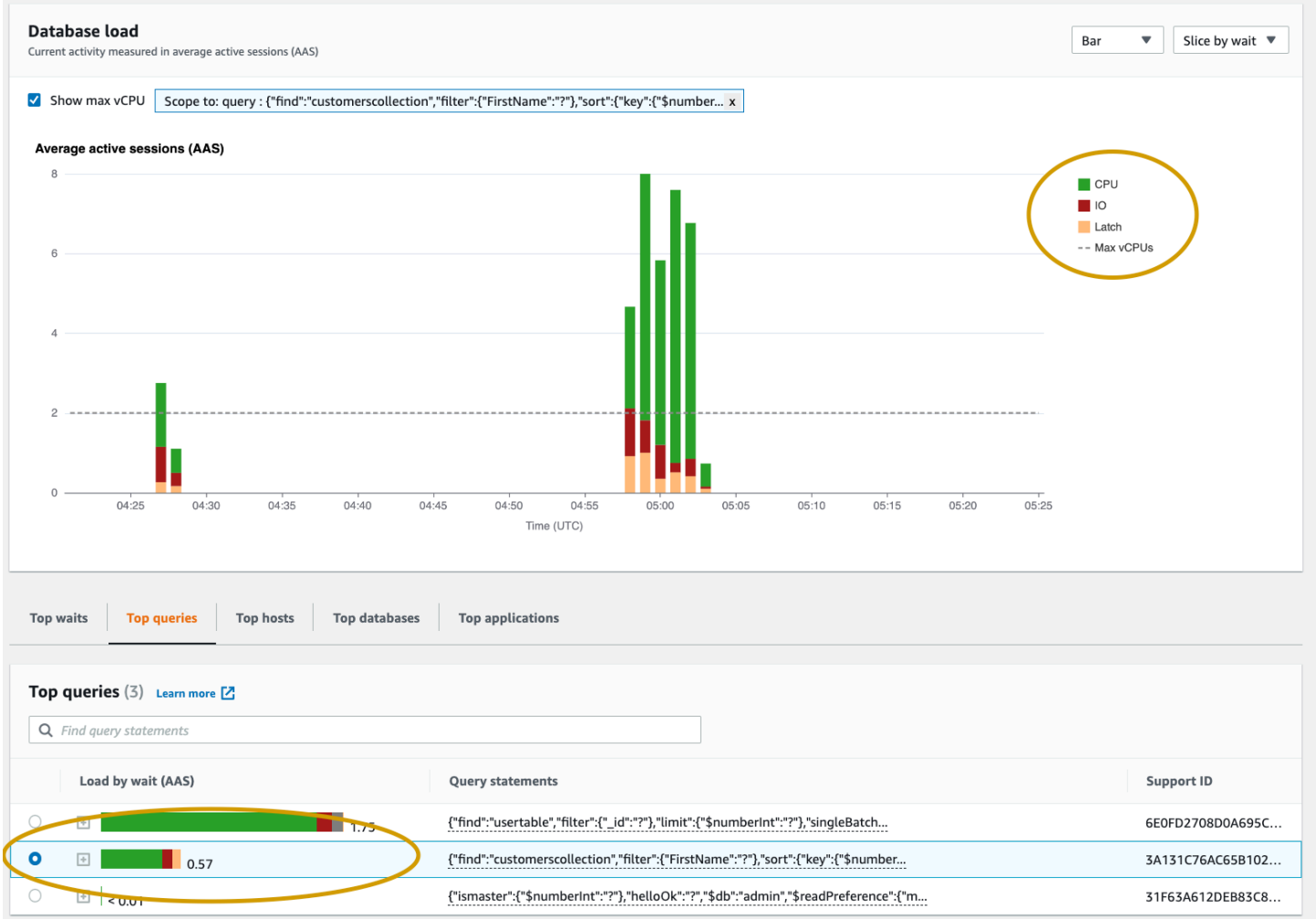
Um die wörtlichen Abfrageanweisungen in einem Digest anzuzeigen, wählen Sie die Abfrage aus und wählen Sie dann das Pluszeichen (). + Im folgenden Screenshot ist die ausgewählte Abfrage ein Digest.



> **Note**
>
> In einem Abfrageüberblick werden ähnliche Abfrageanweisungen gruppiert, vertrauliche Informationen werden jedoch nicht geschwärzt.

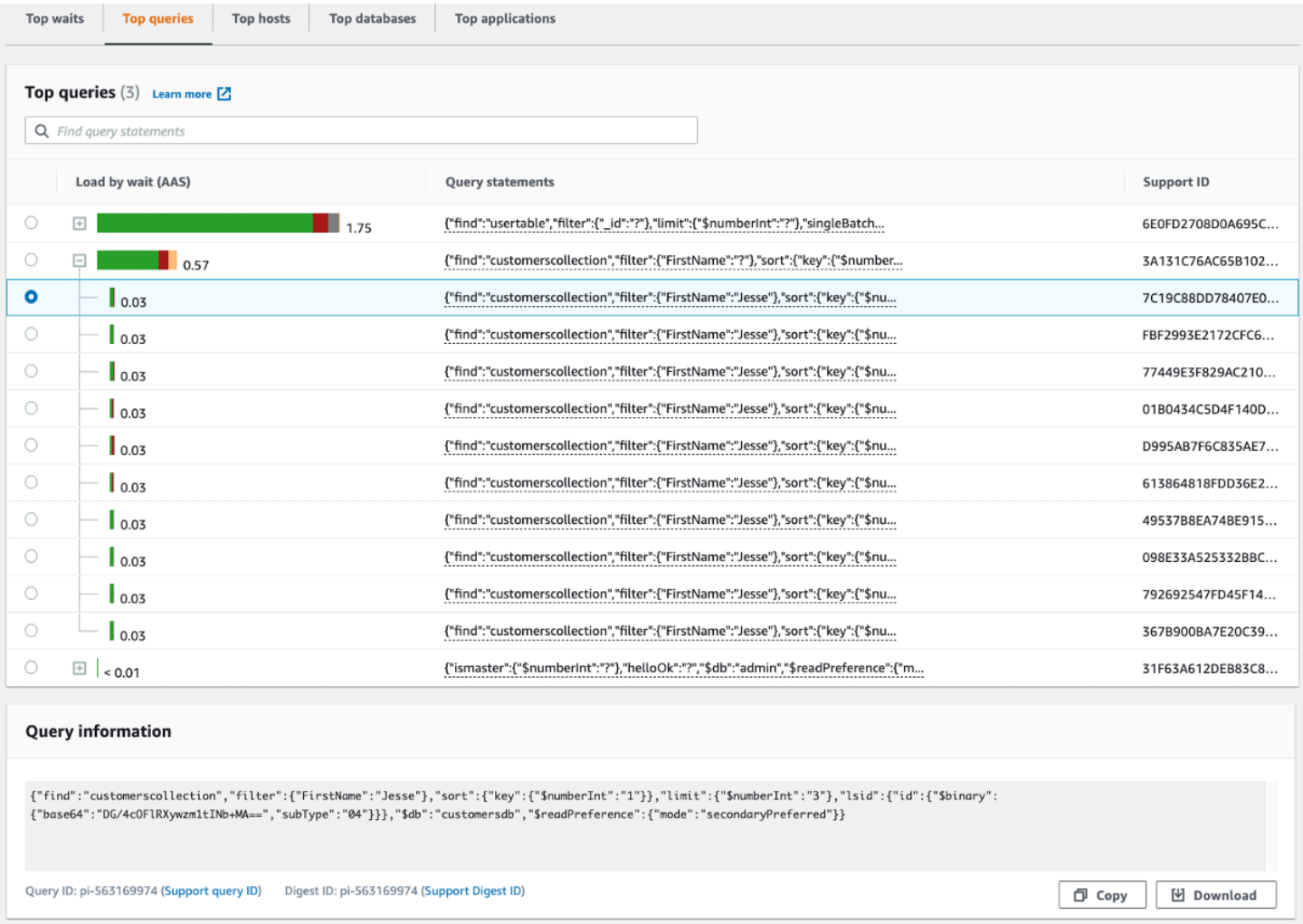
Nach Waits laden (AAS)

Bei Top-Abfragen zeigt die Spalte Load by Waits (AAS) den Prozentsatz der Datenbanklast an, der jedem Top-Load-Element zugeordnet ist. Diese Spalte gibt die Belastung für dieses Element anhand der Gruppierung wieder, die aktuell im DB-Lastdiagramm ausgewählt ist. Beispielsweise können Sie das DB-Last-Diagramm nach Wartezuständen gruppieren. In diesem Fall ist der Balken DB Load by Waits (DB-Last nach Wartezuständen) so groß, segmentiert und farbcodiert, dass angezeigt wird, zu wieviel Prozent diese Abfrage zum betreffenden Wartezustand beiträgt. Es zeigt zudem auf, welche Wartezustände sich auf die ausgewählte Abfrage auswirken.



Anzeige detaillierter Abfrageinformationen

In der Top-Abfragetabelle können Sie eine Digest-Anweisung öffnen, um die zugehörigen Informationen anzuzeigen. Die Informationen werden im unteren Bereich angezeigt.



Die folgenden Typen von Bezeichnern (IDs) sind Abfrageanweisungen zugeordnet:

1. Support-Abfrage-ID — Ein Hashwert der Abfrage-ID. Dieser Wert dient nur zur Referenzierung einer Abfrage-ID, wenn Sie mit AWS Support zusammenarbeiten. AWS Der Support hat keinen Zugriff auf Ihre tatsächlichen Abfrage-IDs und Ihren Abfragetext.
2. Support-Digest-ID — Ein Hashwert der Digest-ID. Dieser Wert dient nur zur Referenzierung einer Digest-ID, wenn Sie mit dem Support zusammenarbeiten. AWS AWS Der Support hat keinen Zugriff auf Ihre tatsächlichen Digest-IDs und Ihren Abfragetext.

Zugriff auf den Abfragetext der Anweisung

Standardmäßig enthält jede Zeile in der Tabelle mit den häufigsten Abfragen 500 Byte an Abfragetext für jede Abfrageanweisung. Wenn eine Digest-Anweisung 500 Byte überschreitet, können Sie mehr Text anzeigen, indem Sie die Anweisung im Performance Insights-Dashboard

öffnen. In diesem Fall beträgt die maximale Länge für die angezeigte Abfrage 1 KB. Wenn Sie eine vollständige Abfrageanweisung anzeigen, können Sie auch Herunterladen wählen.

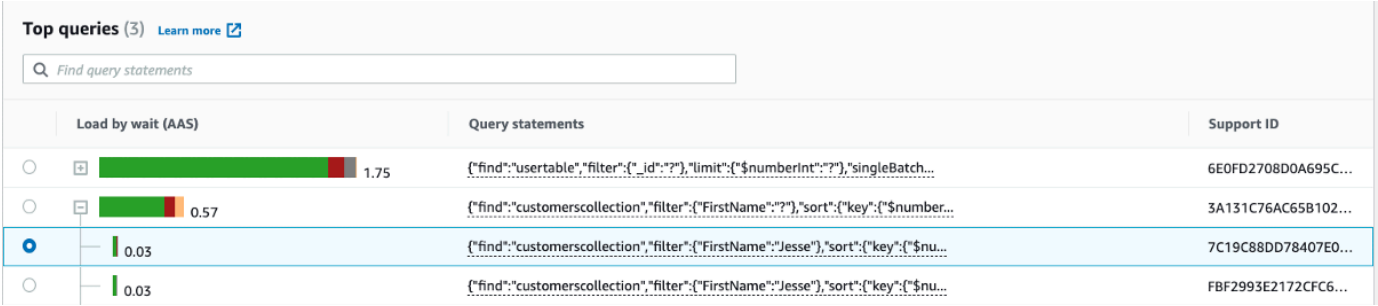
Abfragetext für eine Aussage anzeigen und herunterladen

Im Performance Insights-Dashboard können Sie den Abfragetext anzeigen oder herunterladen.

Um weiteren Abfragetext im Performance Insights-Dashboard anzuzeigen

1. Öffnen Sie die Amazon DocumentDB-Konsole unter: https://console.aws.amazon.com/docdb/
2. Wählen Sie im Navigationsbereich Performance-Insights aus.
3. Wählen Sie eine DB-Instance aus. Das Performance Insights-Dashboard wird für diese DB-Instance angezeigt.

   Abfrageanweisungen mit Text, der größer als 500 Byte ist, sehen wie in der folgenden Abbildung aus:



4. Sehen Sie sich den Abschnitt mit den Abfrageinformationen an, um mehr vom Abfragetext zu sehen.



Das Performance Insights-Dashboard kann bis zu 1 KB für jede vollständige Abfrageanweisung anzeigen.

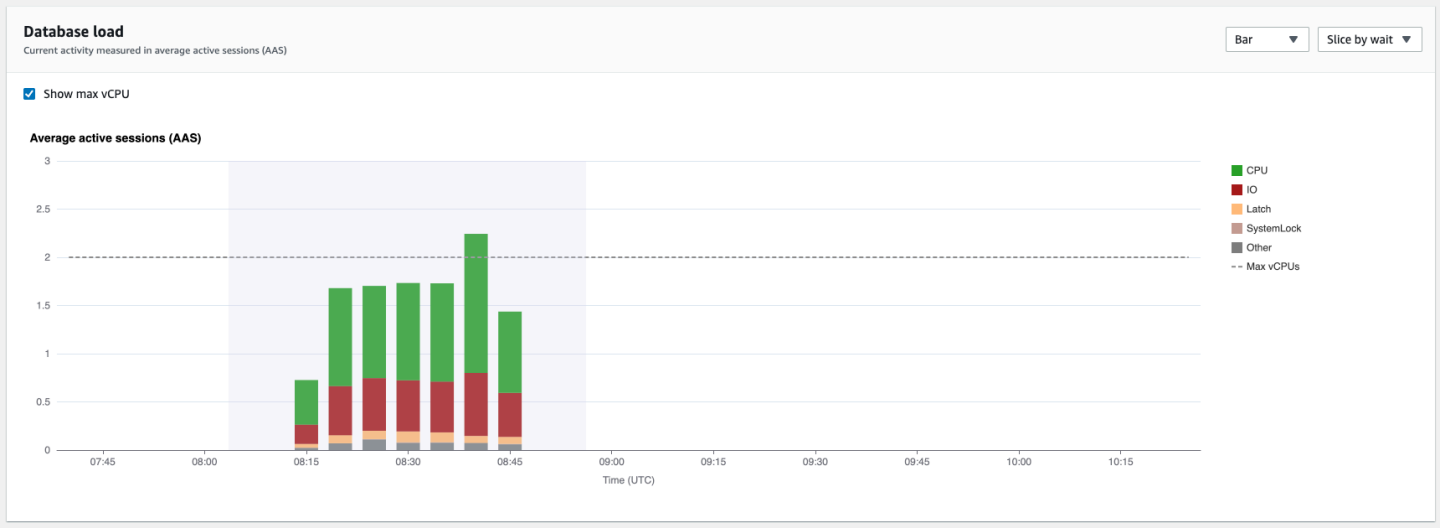
> **Note**
>
> Um die Abfrageanweisung zu kopieren oder herunterzuladen, deaktivieren Sie alle Popup-Blocker.

Das Datenbank-Ladediagramm vergrößern

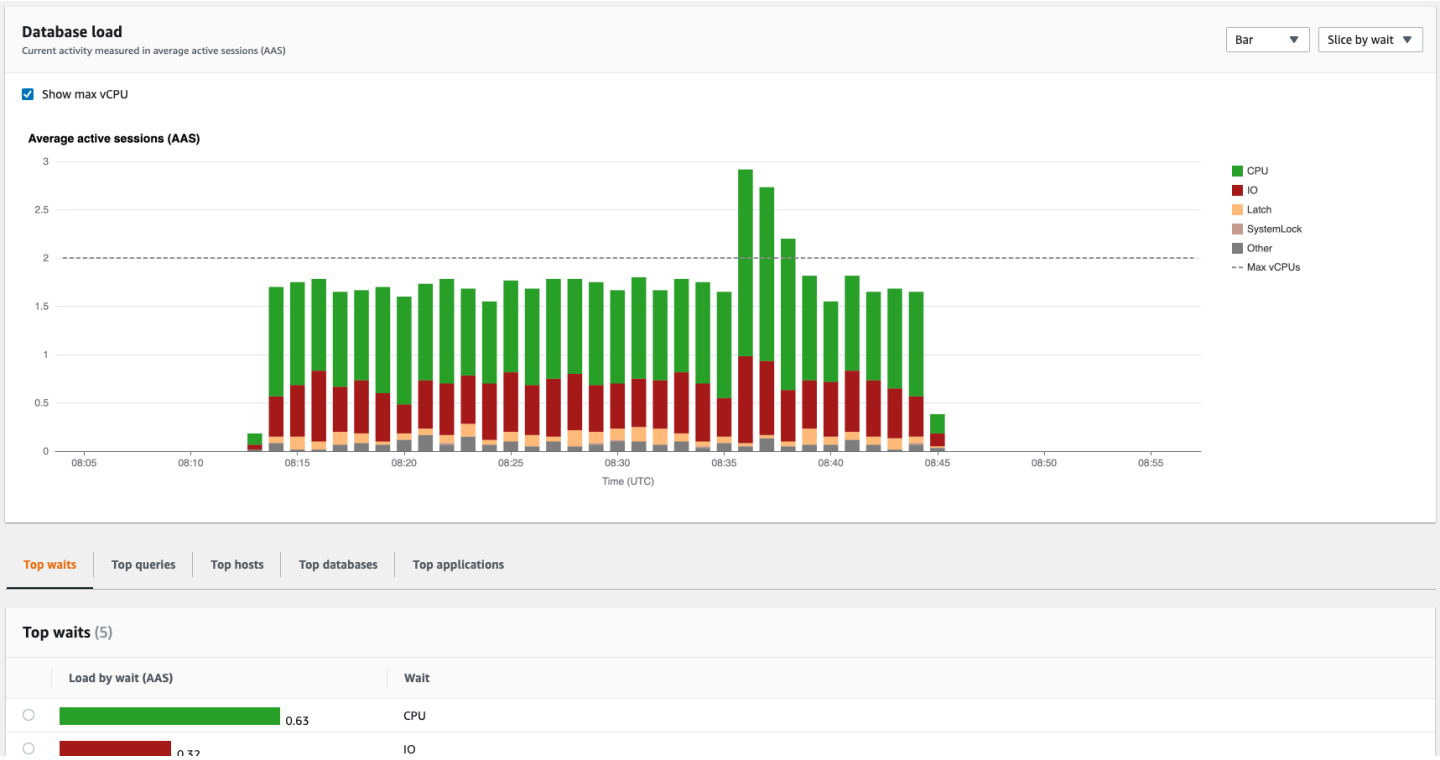
Sie können weitere Funktionen der Benutzeroberfläche von Performance Insights verwenden, um die Performance-Daten zu analysieren.

Click-and-Drag Hineinzoomen

In der Benutzeroberfläche von Performance Insights können Sie einen kleinen Teil des Lastdiagramms auswählen und die Details vergrößern.



Um einen Teil des Lastdiagramms zu vergrößern, wählen Sie die Startzeit und ziehen Sie mit der Maus an das Ende des gewünschten Zeitraums. Dabei wird der markierte Bereich farblich hervorgehoben. Wenn Sie die Maustaste loslassen, vergrößert das Lastdiagramm den ausgewählten Bereich, und die Tabelle mit den wichtigsten Elementen wird neu berechnet.



Abrufen von Metriken mit der Performance-Insights-API

Wenn Performance Insights aktiviert ist, bietet die API Einblicke in die Instance-Leistung Amazon CloudWatch Logs ist die maßgebliche Quelle für Metriken zur Überwachung von Verkäufern für AWS Dienste.

Performance Insights bietet eine domänenspezifische Ansicht der Datenbanklast, gemessen als durchschnittliche aktive Sitzungen (AAS). Diese Metrik erscheint API-Verbrauchern als zweidimensionaler Zeitreihendatensatz. Die Zeitdimension der Daten stellt die Datenbanklastdaten für jeden Zeitpunkt im abgefragten Zeitraum bereit. Für jeden Zeitpunkt wird die Gesamtlast bezogen auf die angeforderten Dimensionen zerlegt, z. B. `Query`, `Wait-state`, `Application` oder `Host`, gemessen zum betreffenden Zeitpunkt.

Amazon DocumentDB Performance Insights überwacht Ihre Amazon DocumentDB-DB-Instance, sodass Sie die Datenbankleistung analysieren und Fehler beheben können. Eine Möglichkeit zum Anzeigen von Performance-Insights-Daten bietet die AWS-Managementkonsole. Performance Insights stellt außerdem eine öffentliche API bereit, sodass Sie Ihre eigenen Daten abfragen können. Sie können die API für Folgendes verwenden:

- Auslagern von Daten in eine Datenbank
- Hinzufügen von Performance-Insights-Daten zu bestehenden Überwachungs-Dashboards

- Entwickeln von Überwachungstools

Um die Performance Insights-API zu verwenden, aktivieren Sie Performance Insights auf einer Ihrer Amazon DocumentDB-Instances. Weitere Informationen zum Aktivieren von Performance Insights finden Sie unter Aktivieren und Deaktivieren von Performance Insights. Weitere Informationen zur Performance Insights-API finden Sie in der Referenz zur Performance Insights-API.

Die Performance-Insights-API bietet die folgenden Operationen.

Performance-Insights-Aktion	AWS CLI Befehl	Description
DescribeDimensionKeys	aws pi describe-dimension-keys	Ruft die Schlüssel der Top N-Dimension für eine Metrik für einen bestimmten Zeitraum ab.
GetDimensionKeyDetails	aws pi get-dimension-key-details	Ruft die Attribute der angegebenen Dimensionsgruppe für eine DB-Instance oder Datenquelle ab. Wenn Sie beispielsweise eine Abfrage-ID angeben und die Dimensionsdetails verfügbar sind, wird der vollständige Text der Dimension `GetDimensionKeyDetails` abgerufen, die dieser ID `db.query.statement` zugeordnet ist. Dieser Vorgang ist nützlich, weil er `GetResourceMetrics` das Abrufen großer Abfrageanweisungstexte `DescribeDimensionKeys` nicht unterstützt.

Performance-Insights-Aktion	AWS CLI Befehl	Description
GetResourceMetadata	aws pi get-resource-metadata	Rufen Sie die Metadaten für verschiedene Funktionen ab. Die Metadaten könnten beispielsweise darauf hindeuten, dass eine Funktion für eine bestimmte DB-Instance ein- oder ausgeschaltet ist.
GetResourceMetrics	aws pi get-resource-metrics	Ruft Performance-Insights-Metriken für eine Reihe von Datenquellen über einen Zeitraum ab. Sie können spezifische Dimensionsgruppen und Dimensionen bereitstellen und Aggregation und Filterkriterien für jede Gruppe bereitstellen.
ListAvailableResourceDimensions	aws pi list-available-resource-dimensions	Rufen Sie die Dimensionen ab, die für jeden angegebenen Metriktyp für eine bestimmte Instance abgefragt werden können.
ListAvailableResourceMetrics	aws pi list-available-resource-metrics	Rufen Sie alle verfügbaren Metriken der angegebenen Metriktypen ab, die für eine bestimmte DB-Instance abgefragt werden können.

Themen

- AWS CLI für Performance Insights
- Abrufen von Zeitreihenmetriken
- AWS CLI Beispiele für Performance Insights

AWS CLI für Performance Insights

Sie können Performance-Insights-Daten über die anzeige AWS CLI. Hilfe zu den AWS CLI -Befehlen für Performance Insights erhalten Sie durch Eingabe der folgenden Befehle an der Befehlszeile.

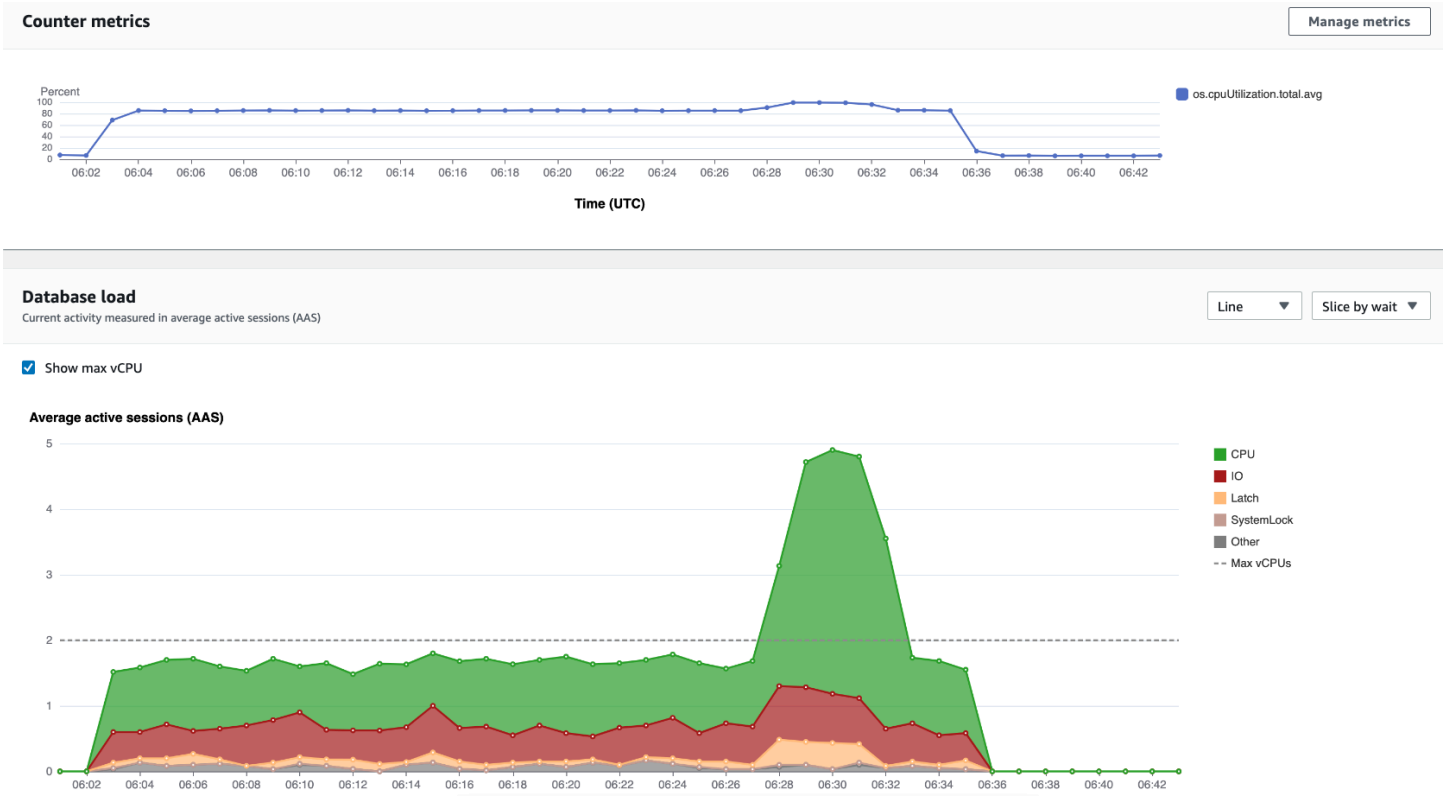
```
aws pi help
```

Wenn Sie das nicht AWS CLI installiert haben, finden Sie im AWS CLIBenutzerhandbuch unter Installation der AWS Befehlszeilenschnittstelle weitere Informationen zur Installation.

Abrufen von Zeitreihenmetriken

Mit der `GetResourceMetrics`-Operation werden ein oder mehrere Zeitreihenmetriken aus den Performance Insights-Daten abgerufen. Für `GetResourceMetrics` ist eine Metrik und ein Zeitraum erforderlich, damit eine Antwort mit einer Liste von Datenpunkten zurückgegeben wird.

Zum Beispiel AWS-Managementkonsole verwendet, `GetResourceMetrics` um das Diagramm Zähler-Metriken und das Diagramm zur Datenbankauslastung auszufüllen, wie in der folgenden Abbildung dargestellt.



Alle von zurückgegebenen Metriken`GetResourceMetrics` sind Standard-Zeitreihenmetriken, mit Ausnahme von`db.load`. Diese Metrik wird im Diagramm Database Load (Datenbanklast)

angezeigt. Die`db.load` Metrik unterscheidet sich von den anderen Zeitreihenmetriken, da Sie sie in Unterkomponenten aufteilen können, die als Dimensionen bezeichnet werden. In der vorherigen Abbildung wird `db.load` unterteilt und nach Wartezuständen gruppiert, aus denen `db.load` besteht.

> **Note**
>
> `GetResourceMetrics` kann auch die `db.sampleload`-Metrik zurückgeben, aber die `db.load`-Metrik ist in den meisten Fällen angemessen.

Informationen zu den Zählermetriken, die von `GetResourceMetrics` zurückgegeben werden, finden Sie unter Leistungseinblicke für Zählerkennzahlen.

Die folgenden Berechnungen werden für die Metriken unterstützt:

- Durchschnitt – Der durchschnittliche Wert für die Metrik über einen bestimmten Zeitraum. Fügen Sie dem Metriknamen `.avg` an.
- Minimum – Der minimale Wert für die Metrik über einen bestimmten Zeitraum. Fügen Sie dem Metriknamen `.min` an.
- Maximum – Der maximale Wert für die Metrik über einen bestimmten Zeitraum. Fügen Sie dem Metriknamen `.max` an.
- Summe – Die Summe der Metrikwerte über einen bestimmten Zeitraum. Fügen Sie dem Metriknamen `.sum` an.
- Beispielanzahl – Die Anzahl, wie oft die Metrik über einen bestimmten Zeitraum erfasst wurde. Fügen Sie dem Metriknamen `.sample_count` an.

Nehmen wir an, dass eine Metrik beispielsweise 300 Sekunden (5 Minuten) lang erfasst wird und dass die Metrik einmal pro Minute erfasst wird. Die Werte für jede Minute sind 1, 2, 3, 4 und 5. In diesem Fall werden die folgenden Berechnungen zurückgegeben:

- Durchschnitt – 3
- Minimum – 1
- Maximum – 5
- Summe – 15
- Beispielanzahl – 5

Hinweise zur Verwendung des `get-resource-metrics` AWS CLI Befehls finden Sie unter [get-resource-metrics].

Geben Sie für die `--metric-queries`-Option eine oder mehrere Abfragen an, um die entsprechenden Ergebnisse zu erhalten. Jede Abfrage besteht aus einem obligatorischen `Metric`- sowie optionalen `GroupBy`- und `Filter`-Parametern. Es folgt ein Beispiel für eine Spezifikation der `--metric-queries`-Option.

```
{
   "Metric": "string",
   "GroupBy": {
     "Group": "string",
     "Dimensions": ["string", ...],
     "Limit": integer
   },
   "Filter": {"string": "string"
     ...}
```

AWS CLI Beispiele für Performance Insights

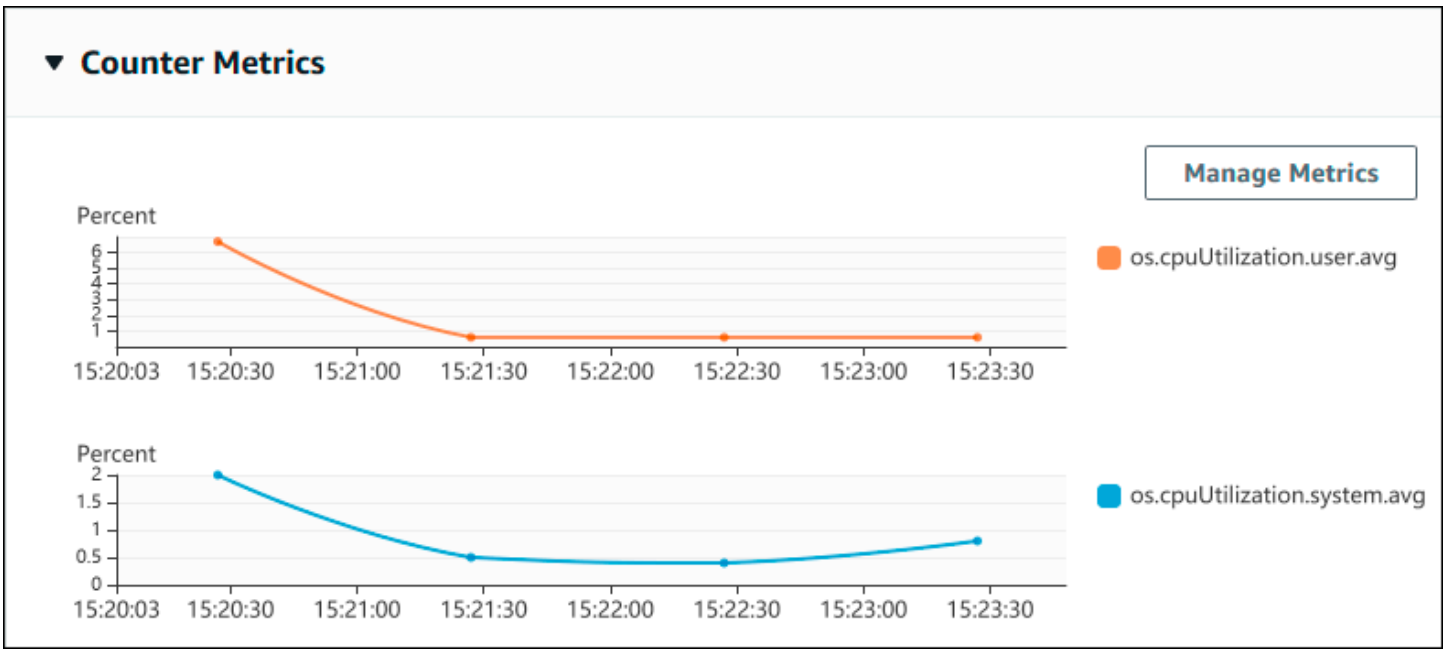
Die folgenden Beispiele zeigen, wie Sie das AWS CLI für Performance Insights verwenden können.

Themen

- Abrufen von Zählermetriken
- Abrufen des Durchschnitts der DB-Auslastung für die Wartezeiten mit den höchsten Wartezeiten
- Abrufen des Durchschnitts der DB-Auslastung für die höchste Abfrage
- Abrufen des nach Query gefilterten Durchschnitts der DB-Auslastung

Abrufen von Zählermetriken

Der folgende Screenshot zeigt zwei Zählermetriken-Diagramme in der AWS-Managementkonsole.



Das folgende Beispiel veranschaulicht, wie die Daten, die die AWS-Managementkonsole zum Erstellen der beiden Zählermetriken-Diagramme verwendet, gesammelt werden.

Für Linux, macOS oder Unix:

```
aws pi get-resource-metrics \
   --service-type DOCDB \
   --identifier db-ID \
   --start-time 2022-03-13T8:00:00Z \
   --end-time   2022-03-13T9:00:00Z \
   --period-in-seconds 60 \
   --metric-queries '[{"Metric": "os.cpuUtilization.user.avg"  },
                      {"Metric": "os.cpuUtilization.idle.avg"}]'
```

Für Windows:

```
aws pi get-resource-metrics ^
   --service-type DOCDB ^
   --identifier db-ID ^
   --start-time 2022-03-13T8:00:00Z ^
   --end-time   2022-03-13T9:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries '[{"Metric": "os.cpuUtilization.user.avg"  },
                      {"Metric": "os.cpuUtilization.idle.avg"}]'
```

Sie können einen Befehl besser lesbar gestalten, indem Sie eine Datei für die Option `--metrics-query` angeben. Im folgenden Beispiel wird eine Datei namens query.json für die Option verwendet. Die Datei enthält Folgendes.

```
[
    {
        "Metric": "os.cpuUtilization.user.avg"
    },
    {
        "Metric": "os.cpuUtilization.idle.avg"
    }
]
```

Führen Sie den folgenden Befehl aus, um die Datei zu verwenden.

Für Linux, macOS oder Unix:

```
aws pi get-resource-metrics \
   --service-type DOCDB \
   --identifier db-ID \
   --start-time 2022-03-13T8:00:00Z \
   --end-time   2022-03-13T9:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

Für Windows:

```
aws pi get-resource-metrics ^
   --service-type DOCDB ^
   --identifier db-ID ^
   --start-time 2022-03-13T8:00:00Z ^
   --end-time   2022-03-13T9:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

Das vorige Beispiel gibt die folgenden Werte für die Optionen an:

- `--service-type`— DOCDB für Amazon DocumentDB
- `--identifier` – Die Ressource-ID für die DB-Instance
- `--start-time` und `--end-time` – Die ISO 8601-Werte `DateTime` für den abzufragenden Zeitraum mit mehreren unterstützten Formaten

Der Abfragezeitraum beträgt eine Stunde:

- `--period-in-seconds` – `60` für eine Abfrage pro Minute
- `--metric-queries` – Ein Array mit zwei Abfragen, jeweils für nur eine Metrik.

  Der Metrikname verwendet Punkte, um die Metrik in eine sinnvolle Kategorie einzustufen, wobei das letzte Element eine Funktion ist. Im Beispiel lautet die Funktion `avg` für jede Abfrage. Wie bei Amazon CloudWatch sind die unterstützten Funktionen`min`, `max`, `total`, und`avg`.

Die Antwort sieht in etwa so aus:

```
{
    "AlignedStartTime": "2022-03-13T08:00:00+00:00",
    "AlignedEndTime": "2022-03-13T09:00:00+00:00",
    "Identifier": "db-NQF3TTMFQ3GTOKIMJODMC3KQQ4",
    "MetricList": [
        {
            "Key": {
                "Metric": "os.cpuUtilization.user.avg"
            },
            "DataPoints": [
                {
                    "Timestamp": "2022-03-13T08:01:00+00:00", //Minute1
                    "Value": 3.6
                },
                {
                    "Timestamp": "2022-03-13T08:02:00+00:00", //Minute2
                    "Value": 2.6
                },
                //.... 60 datapoints for the os.cpuUtilization.user.avg metric
        {
            "Key": {
                "Metric": "os.cpuUtilization.idle.avg"
            },
            "DataPoints": [
                {
                    "Timestamp": "2022-03-13T08:01:00+00:00",
                    "Value": 92.7
                },
                {
                    "Timestamp": "2022-03-13T08:02:00+00:00",
                    "Value": 93.7
```

```
                },
                //.... 60 datapoints for the os.cpuUtilization.user.avg metric
            ]
        }
    ] //end of MetricList
} //end of response
```

Die Antwort enthält Werte für `Identifier`, `AlignedStartTime` und `AlignedEndTime`. Bei einem `--period-in-seconds`-Wert von `60` wurden Start- und Endzeiten auf die Minute ausgerichtet. Wenn der `--period-in-seconds`-Wert `3600` lautet, werden Start- und Endzeiten auf die Stunde ausgerichtet.

Die `MetricList` in der Antwort enthält eine Reihe von Einträgen, und zwar jeweils mit einem `Key`- und einem `DataPoints`-Eintrag. Jeder `DataPoint` verfügt über einen `Timestamp` und einen `Value`. Jede `Datapoints`-Liste enthält 60 Datenpunkte, da die Abfragen eine Stunde lang jede Minute Daten abfragen, und zwar mit den Werten `Timestamp1/Minute1`, `Timestamp2/Minute2` usw. bis `Timestamp60/Minute60`.

Da sich die Abfrage auf zwei verschiedene Zählermetriken bezieht, enthält die -Antwort zwei Element `MetricList`.

Abrufen des Durchschnitts der DB-Auslastung für die Wartezeiten mit den höchsten Wartezeiten

Das folgende Beispiel ist dieselbe Abfrage, die zum Generieren eines gestapelten Flächenliniendiagramms AWS-Managementkonsole verwendet wird. In diesem Beispiel wird der Wert `db.load.avg` für die letzte Stunde abgerufen, wobei die Last nach den sieben wichtigsten Wartezuständen geteilt wurde. Der Befehl ist mit dem Befehl unter identisc Abrufen von Zählermetriken. Die Datei query.json enthält hingegen Folgendes.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_state", "Limit": 7 }
    }
]
```

Führen Sie den folgenden Befehl aus.

Für Linux, macOS oder Unix:

```
aws pi get-resource-metrics \
```

```
   --service-type DOCDB \
   --identifier db-ID \
   --start-time 2022-03-13T8:00:00Z \
   --end-time   2022-03-13T9:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

Für Windows:

```
aws pi get-resource-metrics ^
   --service-type DOCDB ^
   --identifier db-ID ^
   --start-time 2022-03-13T8:00:00Z ^
   --end-time   2022-03-13T9:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

Das Beispiel gibt die Metrik für `db.load.avg` und a `GroupBy` der sieben wichtigsten Wartezustände an. Einzelheiten zu gültigen Werten für dieses Beispiel finden Sie DimensionGroup in der Performance Insights API-Referenz.

Die Antwort sieht in etwa so aus:

```
{
    "AlignedStartTime": "2022-04-04T06:00:00+00:00",
    "AlignedEndTime": "2022-04-04T06:15:00+00:00",
    "Identifier": "db-NQF3TTMFQ3GTOKIMJODMC3KQQ4",
    "MetricList": [
        {//A list of key/datapoints
            "Key": {
                //A Metric with no dimensions. This is the total db.load.avg
                "Metric": "db.load.avg"
            },
            "DataPoints": [
                //Each list of datapoints has the same timestamps and same number of
 items
                {
                    "Timestamp": "2022-04-04T06:01:00+00:00",//Minute1
                    "Value": 0.0
                },
                {
                    "Timestamp": "2022-04-04T06:02:00+00:00",//Minute2
```

```
                    "Value": 0.0
                },
                //... 60 datapoints for the total db.load.avg key
                ]
        },
        {
            "Key": {
                //Another key. This is db.load.avg broken down by CPU
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_state.name": "CPU"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": "2022-04-04T06:01:00+00:00",//Minute1
                    "Value": 0.0
                },
                {
                    "Timestamp": "2022-04-04T06:02:00+00:00",//Minute2
                    "Value": 0.0
                },
                //... 60 datapoints for the CPU key
            ]
        },//... In total we have 3 key/datapoints entries, 1) total, 2-3) Top Wait
 States
    ] //end of MetricList
} //end of response
```

In dieser Antwort gibt es drei Einträge in der`MetricList`. Es gibt einen Eintrag für den Gesamtwert und jeweils drei Einträge für den Wert`db.load.avg`, der nach einem der drei wichtigsten Wartestatus `db.load.avg` aufgeteilt wird. Da es eine Gruppierungsdimension gab (im Gegensatz zum ersten Beispiel), muss es für jede Gruppierung der Metrik einen Schlüssel geben. Für jede Metrik kann nicht nur ein Schlüssel vorhanden sein, wie im Anwendungsfall der Basiszählermetrik.

Abrufen des Durchschnitts der DB-Auslastung für die höchste Abfrage

Das folgende Beispiel `db.wait_state` gruppiert nach den 10 wichtigsten Abfrageanweisungen. Es gibt zwei verschiedene Gruppen für Abfrageanweisungen:

- `db.query`— Die vollständige Abfrageanweisung, z. B. `{"find":"customers","filter":{"FirstName":"Jesse"},"sort":{"key":{"$numberInt":"1"}}}`

- `db.query_tokenized`— Die tokenisierte Abfrageanweisung, z. B. `{"find":"customers","filter":{"FirstName":"?"},"sort":{"key": {"$numberInt":"?"}},"limit":{"$numberInt":"?"}}`

Bei der Analyse der Datenbankleistung kann es nützlich sein, Abfrageanweisungen, die sich nur durch ihre Parameter unterscheiden, als ein einziges Logikelement zu betrachten. In diesem Fall können Sie `db.query_tokenized` beim Abfragen verwenden. Manchmal ist es jedoch sinnvoller`explain()`, vollständige Abfrageanweisungen mit Parametern zu untersuchen, insbesondere wenn Sie daran interessiert sind. Zwischen tokenisierten und vollständigen Abfragen besteht eine Eltern-Kind-Beziehung, wobei mehrere vollständige Abfragen (untergeordnete Abfragen) unter derselben tokenisierten Abfrage (übergeordnetes Objekt) gruppiert sind.

Der Befehl in diesem Beispiel ähnelt dem Befehl unter Abrufen des Durchschnitts der DB-Auslastung für die Wartezeiten mit den höchsten Wartezeiten. Die Datei query.json enthält hingegen Folgendes.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.query_tokenized", "Limit": 10 }
    }
]
```

Im folgenden Beispiel wird verwende `db.query_tokenized`.

Für Linux, macOS oder Unix:

```
aws pi get-resource-metrics \
   --service-type DOCDB \
   --identifier db-ID \
   --start-time 2022-03-13T8:00:00Z \
   --end-time   2022-03-13T9:00:00Z \
   --period-in-seconds 3600 \
   --metric-queries file://query.json
```

Für Windows:

```
aws pi get-resource-metrics ^
   --service-type DOCDB ^
   --identifier db-ID ^
```

```
    --start-time 2022-03-13T8:00:00Z ^
    --end-time   2022-03-13T9:00:00Z  ^
    --period-in-seconds 3600 ^
    --metric-queries file://query.json
```

In diesem Beispiel werden Abfragen über eine Stunde mit einem Zeitraum von einer Minute in Sekunden abgefragt.

Das Beispiel gibt die Metrik für `db.load.avg` und a `GroupBy` der sieben wichtigsten Wartezustände an. Einzelheiten zu gültigen Werten für dieses Beispiel finden Sie DimensionGroup in der Performance Insights API-Referenz.

Die Antwort sieht in etwa so aus:

```
{
    "AlignedStartTime": "2022-04-04T06:00:00+00:00",
    "AlignedEndTime": "2022-04-04T06:15:00+00:00",
    "Identifier": "db-NQF3TTMFQ3GTOKIMJODMC3KQQ4",
    "MetricList": [
        {//A list of key/datapoints
            "Key": {
                "Metric": "db.load.avg"
            },
            "DataPoints": [
                //... 60 datapoints for the total db.load.avg key
                ]
        },
               {
            "Key": {//Next key are the top tokenized queries
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.query_tokenized.db_id": "pi-1064184600",
                    "db.query_tokenized.id": "77DE8364594EXAMPLE",
                    "db.query_tokenized.statement": "{\"find\":\"customers\",\"filter
\":{\"FirstName\":\"?\"},\"sort\":{\"key\":{\"$numberInt\":\"?\"}},\"limit\"
:{\"$numberInt\":\"?\"},\"$db\":\"myDB\",\"$readPreference\":{\"mode\":\"primary\"}}"
                }
            },
            "DataPoints": [
            //... 60 datapoints
            ]
        },
        // In total 11 entries, 10 Keys of top tokenized queries, 1 total key
```

```
    ] //End of MetricList
} //End of response
```

Diese Antwort enthält 11 Einträge in der `MetricList` (insgesamt 1, 10 Top-Token-Abfragen), wobei jeder Eintrag 24 pro Stunde hat. `DataPoints`

Bei tokenisierten Abfragen gibt es in jeder Dimensionsliste drei Einträge:

- `db.query_tokenized.statement`— Die tokenisierte Abfrageanweisung.
- `db.query_tokenized.db_id` — Die synthetische ID, die Performance Insights für Sie generiert. In diesem Beispiel wird die synthetische ID `pi-1064184600` zurückgegeben.
- `db.query_tokenized.id` – Die ID der Abfrage innerhalb von Performance-Insights.

  In der AWS-Managementkonsole wird diese ID als Support-ID bezeichnet. Sie hat diesen Namen, weil es sich bei der ID um Daten handelt, die der AWS Support untersuchen kann, um Ihnen bei der Behebung eines Problems mit Ihrer Datenbank zu helfen. AWS nimmt die Sicherheit und den Datenschutz Ihrer Daten sehr ernst, und fast alle Daten werden bei Ihnen verschlüsselt gespeichert AWS KMS key. Daher AWS kann niemand im Inneren diese Daten einsehen. Im vorherigen Beispiel wird sowohl `tokenized.statement` als auch `tokenized.db_id` verschlüsselt gespeichert. Wenn Sie ein Problem mit Ihrer Datenbank haben, kann Ihnen der AWS Support unter Angabe der Support-ID weiterhelfen.

Beim Abfragen empfiehlt es sich ggf., eine `Group` in `GroupBy` anzugeben. Für eine präzisere Kontrolle der Daten, die zurückgegeben werden, sollten Sie allerdings die Dimensionsliste angeben. Wenn z. B. lediglich eine `db.query_tokenized.statement` erforderlich ist, kann der query.json-Datei ein `Dimensions`-Attribut hinzugefügt werden.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": {
            "Group": "db.query_tokenized",
            "Dimensions":["db.query_tokenized.statement"],
            "Limit": 10
        }
    }
]
```

Abrufen des nach Query gefilterten Durchschnitts der DB-Auslastung

Die entsprechende API-Abfrage in diesem Beispiel ähnelt dem Befehl unter Abrufen des Durchschnitts der DB-Auslastung für die höchste Abfrage. Die Datei query.json enthält hingegen Folgendes.

```
[
 {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_state", "Limit": 5  },
        "Filter": { "db.query_tokenized.id": "AKIAIOSFODNN7EXAMPLE" }
    }
]
```

In dieser Antwort werden alle Werte entsprechend dem Beitrag der tokenisierten Abfrage AKIAIOSFODNN7EXAMPLE gefiltert, der in der Datei query.json angegeben ist. Die Schlüssel folgen möglicherweise auch einer anderen Reihenfolge als eine Abfrage ohne Filter, da sich die fünf wichtigsten Wartezustände auf die gefilterte Abfrage ausgewirkt haben.

CloudWatch Amazon-Metriken für Performance Insights

Performance Insights veröffentlicht automatisch Kennzahlen bei Amazon CloudWatch. Dieselben Daten können in Performance Insights abgefragt werden, aber wenn die Metriken eingegeben sind, ist es CloudWatch einfach, Alarme hinzuzufügen CloudWatch . Die Metriken können auch leicht zu vorhandenen CloudWatch-Dashboards hinzugefügt werden.

Metrik	Beschreibung
DBLoad	Die Anzahl der aktiven Sitzungen für Amazon DocumentDB. In der Regel sind Sie an den Daten für die durchschnittliche Anzahl der aktiven Sitzungen interessiert. Diese Daten werden in Performance Insights als `db.load.avg` abgefragt.
DBLoadCPU	Die Anzahl der aktiven Sitzungen, bei denen der Wartestatustyp CPU ist. In Performance Insights werden diese Daten abgefragt und

Metrik	Beschreibung
	nach dem Wartestatustyp gefiltert. `db.load.avg CPU`
DBLoadNonCPU	Die Anzahl der aktiven Sitzungen, bei denen der Wartestatustyp nicht CPU ist.

> **Note**
>
> Diese Metriken werden CloudWatch nur veröffentlicht, wenn die DB-Instance belastet ist.

Sie können diese Metriken mithilfe der CloudWatch Konsole AWS CLI, der oder der CloudWatch API untersuchen.

Um beispielsweise die statistischen Daten zur Metrik `DBLoad` anzuzeigen, führen Sie den Befehl get-metric-statistics aus.

```
aws cloudwatch get-metric-statistics \
    --region ap-south-1 \
    --namespace AWS/DocDB \
    --metric-name DBLoad  \
    --period 360 \
    --statistics Average \
    --start-time 2022-03-14T8:00:00Z \
    --end-time 2022-03-14T9:00:00Z \
    --dimensions Name=DBInstanceIdentifier,Value=documentdbinstance
```

Dieses Beispiel generiert eine Ausgabe wie die folgende.

```
{
    "Datapoints": [
        {
            "Timestamp": "2022-03-14T08:42:00Z",
            "Average": 1.0,
            "Unit": "None"
        },
        {
            "Timestamp": "2022-03-14T08:24:00Z",
```

```
            "Average": 2.0,
            "Unit": "None"
        },
        {
            "Timestamp": "2022-03-14T08:54:00Z",
            "Average": 6.0,
            "Unit": "None"
        },
        {
            "Timestamp": "2022-03-14T08:36:00Z",
            "Average": 5.7,
            "Unit": "None"
        },
        {
            "Timestamp": "2022-03-14T08:06:00Z",
            "Average": 4.0,
            "Unit": "None"
        },
        {
            "Timestamp": "2022-03-14T08:00:00Z",
            "Average": 5.2,
            "Unit": "None"
        }
    ],
    "Label": "DBLoad"
}
```


Sie können die mathematische `DB_PERF_INSIGHTS` Metrikfunktion in der CloudWatch Konsole verwenden, um Leistungskennzahlen von Amazon DocumentDB Performance Insights abzufragen. Die `DB_PERF_INSIGHTS` Funktion beinhaltet auch die `DBLoad` Metrik in Intervallen von unter einer Minute. Sie können CloudWatch Alarme für diese Metriken einrichten. Weitere Informationen zum Erstellen eines Alarms finden Sie unter Erstellen eines Alarms für Performance Insights-Zählermesswerte aus einer AWS Datenbank.

Weitere Informationen finden Sie CloudWatch unter Was ist Amazon CloudWatch? im CloudWatch Amazon-Benutzerhandbuch.

Leistungseinblicke für Zählerkennzahlen

Zählermesswerte sind Betriebssystemkennzahlen im Performance Insights-Dashboard. Um Leistungsprobleme zu identifizieren und zu analysieren, können Sie Zählermetriken mit der DB-Last korrelieren.

Performance-Insights-Betriebssystemzähler

Die folgenden Betriebssystem-Leistungsindikatoren sind mit Amazon DocumentDB Performance Insights verfügbar.

Zähler	Typ	Metrik
Aktiv	memory	os.memory.active
buffers	memory	os.memory.buffers
cached	memory	os.memory.cached
dirty	memory	os.memory.dirty
free	memory	os.memory.free
inactive	memory	os.memory.inactive
mapped	memory	os.memory.mapped
pageTables	memory	os.memory.pageTables
slab	memory	os.memory.slab
total	memory	os.memory.total
writeback	memory	os.memory.writeback
idle	cpuUtilization	os.cpu Utilization.idle
system	cpuUtilization	os.cpu Utilization.system
total	cpuUtilization	os.cpu Utilization.total

Zähler	Typ	Metrik
user	cpuUtilization	os.cpu Utilization.user
wait	cpuUtilization	os.cpu Utilization.wait
one	laden AverageMinute	os.load AverageMinute.one
fifteen	laden AverageMinute	os.load AverageMinute.fifteen
fünf	laden AverageMinute	os.load AverageMinute.five
cached	swap	os.swap.cached
free	swap	os.swap.free
in	swap	os.swap.in
out	swap	os.swap.out
total	swap	os.swap.total
rx	network	os.network.rx
tx	network	os.network.tx
numVCPUs	general	os.general.numVCPUs

Serverlose Verwendung von Amazon DocumentDB

Amazon DocumentDB Serverless ist eine On-Demand-Konfiguration mit automatischer Skalierung, die die Prozesse der dynamischen Anpassung der Kapazität Ihrer Amazon DocumentDB-Datenbank je nach Anwendungsbedarf automatisiert. Ihnen werden nur die Ressourcen in Rechnung gestellt, die Ihre Cluster verbrauchen. Somit kann DocumentDB Serverless Ihnen helfen, das Budget einzuhalten und zu vermeiden, für Computerressourcen zu bezahlen, die Sie nicht nutzen.

Diese Art der Automatisierung ist besonders nützlich für Umgebungen mit sehr variablen und unvorhersehbaren Arbeitslasten, wie z. B. mehrinstanzenfähige Datenbanken, verteilte Datenbanken sowie Entwicklungs- und Testsysteme.

Topics

- Anwendungsfälle für serverlose DocumentDB
- Vorteile von Amazon DocumentDB serverless
- So funktioniert Amazon DocumentDB serverless
- Anforderungen und Einschränkungen für DocumentDB serverless
- Einen Cluster erstellen, der Amazon DocumentDB serverlos verwendet
- Migration zu Amazon DocumentDB serverlos
- Verwaltung von Amazon DocumentDB serverlos
- Grenzwerte für serverlose Amazon DocumentDB-Instances
- Serverlose Skalierungskonfiguration von Amazon DocumentDB
- Überwachung von Amazon DocumentDB (serverlos)

Anwendungsfälle für serverlose DocumentDB

Sowohl von Amazon DocumentDB bereitgestellte Cluster als auch DocumentDB serverless unterstützen viele Arten von Datenbank-Workloads, von Entwicklungs- und Testumgebungen bis hin zu den anspruchsvollsten, geschäftskritischen Anwendungen, die eine hohe Skalierung und Verfügbarkeit erfordern. DocumentDB Serverless erweitert die Workloads der Kunden jedoch um eine weitere Dimension, nämlich die Fähigkeit, Websites und Anwendungen mit unvorhersehbaren Arbeitslasten zu unterstützen.

DocumentDB Serverless ist besonders für die folgenden Anwendungsfälle nützlich:

- Variable Workloads — Sie führen Workloads aus, bei denen die Aktivität plötzlich und unvorhersehbar zunimmt. Ein Beispiel dafür ist eine Verkehrs-Website, für die Aktivitätsspitzen entstehen, wenn es zu regnen beginnt. Eine andere ist eine E-Commerce-Website mit erhöhtem Datenverkehr, wenn Sie Verkaufs- oder Sonderaktionen anbieten. Mit DocumentDB serverless skaliert Ihre Datenbank automatisch die Kapazität, um den Anforderungen der Spitzenlast der Anwendung gerecht zu werden, und wird wieder herunterskaliert, wenn der Aktivitätsschub vorbei ist. Mit DocumentDB serverless müssen Sie nicht mehr für Spitzen- oder Durchschnittskapazitäten sorgen. Sie können eine obere Kapazitätsgrenze angeben, um die schlimmste Situation zu bewältigen. Diese Kapazität wird nur genutzt, wenn sie benötigt wird.
  - Die Granularität der Skalierung in DocumentDB Serverless hilft Ihnen, die Kapazität genau an die Anforderungen Ihrer Datenbank anzupassen. Bei einem bereitgestellten Cluster erfordert die Hochskalierung das Hinzufügen einer völlig neuen Instanz. DocumentDB Serverless kann eine halbe DCU hinzufügen, wenn nur etwas mehr Kapazität benötigt wird. Je nach der zusätzlichen Kapazität, die zur Bewältigung einer erhöhten Arbeitslast erforderlich ist, können 0,5, 1, 1,5, 2 oder weitere halbe DCU hinzugefügt werden. Und es kann 0,5, 1, 1,5, 2 oder weitere halbe DCU entfernen, wenn die Arbeitslast abnimmt und diese Kapazität nicht mehr benötigt wird.
- Multi-tenant Anwendungen — Mit DocumentDB serverless müssen Sie die Datenbankkapazität nicht für jede Anwendung in Ihrer Flotte einzeln verwalten. DocumentDB Serverless verwaltet die individuelle Datenbankkapazität für Sie.
  - Sie können einen Cluster für jeden Mandanten erstellen. Auf diese Weise können Sie Funktionen wie Klonen und Snapshot-Wiederherstellung verwenden, um die Hochverfügbarkeit und die Notfallwiederherstellung je nach Mandant zu verbessern.
  - Jeder Mandant kann je nach Tageszeit, Jahreszeit, Werbeveranstaltungen usw. bestimmte Perioden mit hoher und geringer Aktivität haben. Jeder Cluster kann einen großen Kapazitätsbereich aufweisen. Auf diese Weise fallen für Cluster mit geringer Aktivität nur minimale Instance-Gebühren an. Jeder Cluster kann schnell hochskaliert werden, um Perioden mit hoher Aktivität zu bewältigen.
- Neue Anwendungen — Sie stellen eine neue Anwendung bereit und sind sich nicht sicher, welche Instanzgröße Sie benötigen. Wenn Sie DocumentDB serverless verwenden, können Sie einen Cluster mit einer oder mehreren Instanzen einrichten und die Datenbank automatisch an die Kapazitätsanforderungen Ihrer Anwendung anpassen lassen.
- Mixed-use Anwendungen — Nehmen wir an, Sie haben eine OLTP-Anwendung (Online Transaction Processing), aber es kommt regelmäßig zu Spitzen im Abfrageverkehr. Indem Sie Aufstiegsstufen für die serverlosen DocumentDB-Instances in einem Cluster angeben, können Sie Ihren Cluster so konfigurieren, dass die Reader-Instances unabhängig von der Writer-Instance

skaliert werden können, um die zusätzliche Last zu bewältigen. Wenn die Auslastungsspitze nachlässt, werden die Reader-Instances wieder herunterskaliert, um der Kapazität der Writer-Instanz zu entsprechen.

- Kapazitätsplanung — Nehmen wir an, Sie passen in der Regel Ihre Datenbankkapazität an oder überprüfen die optimale Datenbankkapazität für Ihre Arbeitslast, indem Sie die Instance-Klassen aller Instances in einem Cluster ändern. Mit DocumentDB serverlos können Sie diesen Verwaltungsaufwand vermeiden. Sie können die angemessene Mindest- und Höchstkapazität ermitteln, indem Sie den Workload ausführen und überprüfen, wie stark die Instances tatsächlich skaliert werden.
  - Sie können vorhandene Instanzen von „Provisioned“ zu „DocumentDB serverless“ oder von „DocumentDB serverless“ zu „Provisioned“ ändern. In solchen Fällen müssen Sie weder einen neuen Cluster noch eine neue Instanz erstellen.
- Entwicklung und Testen — Sie können DocumentDB nicht nur für Ihre anspruchsvollsten Anwendungen ausführen, sondern auch serverlos für Entwicklungs- und Testumgebungen verwenden. Mit DocumentDB serverless können Sie Instances mit einer niedrigen Mindestkapazität erstellen, anstatt burstable-db.t*-Instance-Klassen zu verwenden. Sie können die maximale Kapazität so hoch einstellen, dass diese Instances immer noch umfangreiche Workloads ausführen können, ohne dass der Arbeitsspeicher knapp wird. Wenn die Datenbank nicht verwendet wird, werden alle Instances herunterskaliert, um unnötige Kosten zu vermeiden.

Verwendung von Amazon DocumentDB serverless für bestehende bereitgestellte Workloads

Angenommen, Sie haben bereits eine DocumentDB-Anwendung, die auf einem bereitgestellten Cluster ausgeführt wird. Sie können überprüfen, wie die Anwendung mit DocumentDB serverless funktionieren würde, indem Sie dem vorhandenen Cluster eine oder mehrere serverlose DocumentDB-Instanzen als Reader-Instances hinzufügen. Sie können überprüfen, wie oft die Reader-Instanzen nach oben und unten skaliert werden. Sie können den DocumentDB-Failover-Mechanismus verwenden, um eine serverlose DocumentDB-Instanz zum Writer zu machen und zu überprüfen, wie sie mit der Arbeitslast umgeht. read/write Auf diese Weise können Sie mit minimalen Ausfallzeiten und ohne Ändern des Endpunkts, den Ihre Clientanwendungen verwenden, umstellen. Einzelheiten zum Verfahren zur Konvertierung vorhandener Cluster in serverlose DocumentDB-Cluster finden Sie unter. Migration zu Amazon DocumentDB serverlos

Vorteile von Amazon DocumentDB serverless

DocumentDB serverless ist für variable oder „sprunghafte“ Workloads vorgesehen. Bei solch unvorhersehbaren Workloads haben Sie möglicherweise Schwierigkeiten zu planen, wann Ihre Datenbankkapazität geändert werden soll. Möglicherweise haben Sie auch Probleme, Kapazitätsänderungen mithilfe der vertrauten Mechanismen wie dem Hinzufügen von Instanzen oder dem Ändern von Instanzklassen schnell genug vorzunehmen. DocumentDB Serverless bietet die folgenden Vorteile, um in solchen Anwendungsfällen zu helfen:

- Einfacheres Kapazitätsmanagement als bereitgestellt — DocumentDB Serverless reduziert den Aufwand für die Planung der Instanzgrößen und die Größenänderung von Instanzen, wenn sich die Arbeitslast ändert. Es reduziert auch den Aufwand für die Aufrechterhaltung einer konsistenten Kapazität für alle Instanzen in einem Cluster.

- Schnellere und einfachere Skalierung in Zeiten hoher Aktivität — DocumentDB serverless skaliert die Rechen- und Speicherkapazität nach Bedarf, ohne dass die Kundentransaktionen oder Ihre gesamte Arbeitslast unterbrochen werden. Die Möglichkeit, Reader-Instances mit DocumentDB serverless zu verwenden, hilft Ihnen, zusätzlich zur vertikalen Skalierung die Vorteile der horizontalen Skalierung zu nutzen. Die Möglichkeit, globale DocumentDB-Cluster zu verwenden, bedeutet, dass Sie Ihre serverlose DocumentDB-Lese-Arbeitslast auf mehrere verteilen können. AWS-Regionen Diese Fähigkeit ist bequemer als die Skalierungsmechanismen für bereitgestellte Cluster.

- Cost-effective in Zeiten geringer Aktivität — DocumentDB Serverless hilft Ihnen dabei, eine Überversorgung Ihrer Instances zu vermeiden. DocumentDB Serverless fügt Ressourcen in granularen Schritten hinzu, wenn die Instances skaliert werden. Sie zahlen nur für die Datenbankressourcen, die Sie verbrauchen. Die serverlose Nutzung der DocumentDB-Ressourcen wird pro Sekunde gemessen. Auf diese Weise wird der reduzierte Ressourcenverbrauch sofort registriert, wenn eine Instanz herunterskaliert wird.

- Funktionsparität mit Provisioned — Sie können alle DocumentDB-Funktionen mit DocumentDB serverless verwenden. Mit DocumentDB Serverless können Sie beispielsweise Reader-Instances, AWS Identity and Access Management (IAM) -Datenbankauthentifizierung und Performance Insights verwenden.

Mit DocumentDB Serverless können Sie insbesondere die folgenden Funktionen der bereitgestellten Cluster nutzen:

- Reader-Instances — DocumentDB Serverless kann Reader-Instances nutzen, um horizontal zu skalieren. Wenn ein Cluster eine oder mehrere Reader-Instanzen enthält, kann es bei Problemen mit der Writer-Instanz sofort zu einem Failover des Clusters kommen.
- Multi-AZ Cluster — Sie können die serverlosen DocumentDB-Instanzen eines Clusters auf mehrere Availability Zones (AZs) verteilen. Die Einrichtung eines Multi-AZ Clusters trägt dazu bei, die Geschäftskontinuität auch im seltenen Fall von Problemen sicherzustellen, die eine gesamte AZ betreffen.
- Globale Cluster — Sie können DocumentDB serverless in Kombination mit globalen DocumentDB-Clustern verwenden, um zusätzliche schreibgeschützte Kopien Ihres Clusters in anderen AWS-Regionen Clustern für Disaster Recovery-Zwecke zu erstellen.

So funktioniert Amazon DocumentDB serverless

Topics

- -Übersicht
- Konfigurationen für Amazon DocumentDB-Cluster
- Serverlose Skalierungskapazität von Amazon DocumentDB
- Serverlose Skalierung von Amazon DocumentDB
- Ruhezustand (0,5 DCUs)

-Übersicht

Amazon DocumentDB Serverless eignet sich für die anspruchsvollsten, sehr variablen Workloads. Zum Beispiel könnte Ihre Datenbanknutzung für einen kurzen Zeitraum sehr hoch sein, gefolgt von langen Zeiträumen mit nur leichter oder gar keiner Aktivität. Beispiele hierfür sind Einzelhandels-Websites, Spiele oder Sportveranstaltungen mit regelmäßigen Werbeveranstaltungen sowie Datenbanken, die nach Bedarf Berichte erstellen. Andere Beispiele sind Entwicklungs- und Testumgebungen sowie neue Anwendungen, in denen die Nutzung schnell ansteigen könnte. In Fällen wie diesen und vielen anderen ist es mit dem bereitgestellten Modell nicht immer möglich, die Kapazität vorab korrekt zu konfigurieren. Es können auch höhere Kosten entstehen, wenn Sie zu viel Kapazität bereitstellen, die Sie dann nicht benötigen.

Im Gegensatz dazu eignen sich von DocumentDB bereitgestellte Cluster für stetige Workloads. Bei bereitgestellten Clustern wählen Sie eine Instanzklasse aus, die über eine vordefinierte Menge an Arbeitsspeicher, CPU-Leistung, I/O Bandbreite usw. verfügt. Wenn sich Ihre Workload ändert, ändern Sie die Instance-Klasse Ihres Writers und Ihrer Readers manuell. Das bereitgestellte Modell funktioniert gut, wenn Sie die Kapazität im Vorfeld der erwarteten Verbrauchsmustern anpassen können. Es ist akzeptabel, dass kurze Ausfälle auftreten, während Sie die Instance-Klasse des Writers und der Reader in Ihrem Cluster ändern.

DocumentDB Serverless wurde von Grund auf so konzipiert, dass serverlose Cluster unterstützt werden, die sofort skalierbar sind. DocumentDB Serverless wurde so konzipiert, dass es das gleiche Maß an Sicherheit und Isolierung bietet wie bereitgestellte Writer- und Lesegeräte. Diese Aspekte sind in serverlosen Cloud-Umgebungen mit mehreren Mandanten entscheidend. Der dynamische Skalierungsmechanismus umfasst sehr wenig Aufwand, sodass er schnell auf Änderungen der Datenbank-Workload reagieren kann. Er bietet auch genügend Leistung, um einen drastischen Anstieg beim Bedarf nach Rechenleistung zu meistern.

Wenn Sie DocumentDB serverless verwenden, können Sie einen DocumentDB-Cluster erstellen, ohne für jeden Writer und Reader an eine bestimmte Datenbankkapazität gebunden zu sein. Sie geben den minimalen und maximalen Bereich für die Kapazität an. DocumentDB skaliert jeden serverlosen DocumentDB-Writer oder -Leser im Cluster innerhalb dieses Kapazitätsbereichs. Durch die Verwendung eines Multi-AZ Clusters, in dem jeder Writer oder Reader dynamisch skalieren kann, können Sie die Vorteile der dynamischen Skalierung und Hochverfügbarkeit nutzen.

DocumentDB serverless skaliert die Datenbankressourcen automatisch auf der Grundlage Ihrer Mindest- und Höchstkapazitätsspezifikationen. Die Skalierung ist schnell, da die meisten Skalierungsereignisoperationen den Writer oder Reader auf demselben Host halten. In den seltenen Fällen, in denen ein serverloser DocumentDB-Writer oder -Leser von einem Host auf einen anderen verschoben wird, verwaltet DocumentDB Serverless die Verbindungen automatisch. Sie müssen Ihren Datenbank-Client-Anwendungscode oder Ihre Datenbank-Verbindungszeichenfolgen nicht ändern.

Bei DocumentDB Serverless sind wie bei bereitgestellten Clustern Speicherkapazität und Rechenkapazität getrennt. Wenn wir von serverloser DocumentDB-Kapazität und Skalierung sprechen, ist es immer die Rechenkapazität, die zunimmt oder sinkt. Somit kann Ihr Cluster viele Terabyte an Daten enthalten, auch wenn die CPU und Speicherkapazität auf ein niedriges Niveau herunterskalieren.

Anstelle der Bereitstellung und Verwaltung von Datenbankservern geben Sie die Datenbankkapazität an. Die tatsächliche Kapazität jedes serverlosen DocumentDB-Schreibers oder -Lesegeräts variiert

im Laufe der Zeit, abhängig von Ihrer Arbeitslast. Details zu diesem Mechanismus finden Sie unter Serverlose Skalierung von Amazon DocumentDB.

Konfigurationen für Amazon DocumentDB-Cluster

Für jeden Ihrer Amazon DocumentDB-Cluster können Sie eine beliebige Kombination aus serverloser DocumentDB-Kapazität, bereitgestellter Kapazität oder beidem wählen.

Sie können einen Cluster einrichten, der sowohl serverlose als auch bereitgestellte DocumentDB-Kapazität enthält. Dies wird als Cluster mit gemischter Konfiguration bezeichnet. Angenommen, Sie benötigen mehr read/write Kapazität, als für einen serverlosen DocumentDB-Writer verfügbar ist. In diesem Fall können Sie den Cluster mit einem sehr großen bereitgestellten Writer einrichten. Dann können Sie DocumentDB Serverless immer noch für die Reader verwenden. Oder nehmen Sie an, dass die Schreib-Workload für Ihren Cluster variiert, die Lese-Workload jedoch stabil ist. In diesem Fall können Sie Ihren Cluster mit einem serverlosen DocumentDB-Writer und einem oder mehreren bereitgestellten Readern einrichten.

Sie können auch einen Cluster einrichten, in dem die gesamte Kapazität von DocumentDB serverless verwaltet wird. Dazu können Sie einen neuen Cluster erstellen und DocumentDB serverless von Anfang an verwenden. Oder Sie können die gesamte bereitgestellte Kapazität in einem vorhandenen Cluster durch DocumentDB serverless ersetzen. Die Verfahren zum Erstellen eines neuen Clusters mit DocumentDB serverless oder zum Umstellen eines vorhandenen Clusters auf serverlose DocumentDB finden Sie unter und. Einen Cluster erstellen, der Amazon DocumentDB serverlos verwendet Migration zu Amazon DocumentDB serverlos

Wenn Sie DocumentDB Serverless in einem Cluster überhaupt nicht verwenden, werden alle Writer und Reader im Cluster bereitgestellt. Dies ist die gängigste Art von Cluster, mit der die meisten Benutzer vertraut sind. Die bereitgestellte Kapazität ist konstant. Die Gebühren sind relativ einfach zu prognostizieren. Sie müssen jedoch im Vorfeld prognostizieren, wie viel Kapazität Sie benötigen. In einigen Fällen sind Ihre Prognosen möglicherweise ungenau oder Ihr Kapazitätsbedarf kann sich ändern. In diesen Fällen kann es zu einer Unterversorgung (langsamer als gewünscht) oder zu einer Überversorgung (teurer als gewünscht) kommen.

Serverlose Skalierungskapazität von Amazon DocumentDB

Die Maßeinheit für Amazon DocumentDB serverless ist die DocumentDB Capacity Unit (DCU). Die serverlose Skalierungskapazität von DocumentDB ist nicht an die Instance-Klassen gebunden, die Sie für bereitgestellte Cluster verwenden.

Jede DCU ist eine Kombination aus etwa 2 Gibibyte (GiB) Arbeitsspeicher, der entsprechenden CPU und dem Netzwerk. Mit dieser Maßeinheit geben Sie den Kapazitätsbereich der Datenbank an. Die Metriken `ServerlessDatabaseCapacity` und `DCUUtilization` CloudWatch helfen Ihnen festzustellen, wie viel Kapazität Ihre Datenbank tatsächlich nutzt und wo diese Kapazität innerhalb des angegebenen Bereichs liegt.

Zu jedem Zeitpunkt hat jeder serverlose DocumentDB-Writer oder -Leser eine Kapazität. Die Kapazität ist eine Gleitkommazahl, die DCU repräsentiert. Die Kapazität steigt oder nimmt ab, sobald der Writer oder Reader skaliert. Dieser Wert wird jede Sekunde gemessen. Für jeden Cluster, in dem Sie DocumentDB serverless verwenden möchten, definieren Sie einen Kapazitätsbereich: die Mindest- und Höchstkapazitätswerte, zwischen denen jeder serverlose DocumentDB-Writer oder Reader skalieren kann. Der Kapazitätsbereich ist für jeden serverlosen DocumentDB-Writer oder -Leser in einem Cluster derselbe. Jeder serverlose DocumentDB-Writer oder -Leser hat seine eigene Kapazität, die irgendwo in diesem Bereich liegt.

DocumentDB Serverless wird auf DocumentDB 5.0 und 8.0 mit einem Kapazitätsbereich von 0,5 bis 256 DCUs unterstützt.

Die kleinste serverlose DocumentDB-Kapazität, die Sie definieren können, beträgt 0,5 DCUs. Sie können einen höheren Wert angeben, wenn dieser kleiner als oder gleich dem maximal unterstützten Kapazitätswert ist. Wenn Sie die Mindestkapazität auf einen kleinen Wert festlegen, verbrauchen leicht ausgelastete Cluster nur minimale Rechenressourcen. Gleichzeitig bleiben sie bereit, Verbindungen sofort anzunehmen und hochzuskalieren, wenn ihre Aktivität ansteigt.

Setzen Sie den Mindestwert auf einen Wert, der es jedem Schreib- oder Lesegerät ermöglicht, den Arbeitssatz der Anwendung im Pufferpool zu speichern. Auf diese Weise wird der Inhalt des Pufferpools in Zeiten geringer Aktivität nicht verworfen. Alle Überlegungen zur Auswahl des Skalierungskapazitätsbereichs finden Sie unterAuswahl des Skalierungskapazitätsbereichs für einen serverlosen DocumentDB-Cluster.

Je nachdem, wie Sie die Lesegeräte in einer Multi-AZ Bereitstellung konfigurieren, können ihre Kapazitäten an die Kapazität des Schreibers gebunden oder unabhängig voneinander sein. Weitere Informationen zur Vorgehensweise finden Sie unter Die Angebotsstufe für serverlose Lesegeräte anzeigen und ändern.

Bei der serverlosen Überwachung von DocumentDB werden die Kapazitätswerte für Writer und Reader in Ihrem Cluster im Laufe der Zeit gemessen. Wenn Ihre Datenbank nicht auf die Mindestkapazität herunterskaliert, können Sie Maßnahmen ergreifen, z. B. den minimalen Wert anpassen und Ihre Datenbankanwendung optimieren. Wenn Ihre Datenbank ihre maximale Kapazität

konsequent erreicht, können Sie Maßnahmen wie die Erhöhung des maximalen Werts ergreifen. Sie können auch Ihre Datenbankanwendung optimieren und die Abfragelast auf mehr Reader verteilen.

Die Gebühren für die serverlose DocumentDB-Kapazität werden in folgenden Werten gemessen. DCU-hours Informationen darüber, wie die Gebühren für serverlose DocumentDB berechnet werden, finden Sie unter Amazon DocumentDB-Preise. Nehmen wir an, dass die Gesamtzahl der Writer und Reader in Ihrem Cluster n beträgt. In diesem Fall verbraucht der Cluster mindestens etwa n x DCUs, wenn Sie keine Datenbankoperationen ausführen. Amazon DocumentDB selbst führt möglicherweise Überwachungs- oder Wartungsvorgänge durch, die eine geringe Belastung verursachen. Dieser Cluster verbraucht maximal n x DCUs, wenn die Datenbank mit voller Kapazität läuft.

Weitere Informationen zur Auswahl geeigneter Mindest- und Höchstwerte für DCU finden Sie unter. Auswahl des Skalierungskapazitätsbereichs für einen serverlosen DocumentDB-Cluster Die von Ihnen angegebenen Mindest- und Höchstwerte für DCU wirken sich auch auf einige Amazon DocumentDB-Instance-Grenzwerte aus. Einzelheiten zur Interaktion zwischen dem Kapazitätsbereich und den Instance-Limits finden Sie unter. Grenzwerte für serverlose Amazon DocumentDB-Instances

Serverlose Skalierung von Amazon DocumentDB

Für jeden serverlosen DocumentDB-Writer oder -Leser verfolgt Amazon DocumentDB kontinuierlich die Auslastung von Ressourcen wie CPU, Speicher und Netzwerk. Diese Messungen werden zusammen als Last bezeichnet. Die Last umfasst die von Ihrer Anwendung ausgeführten Datenbankoperationen. Es umfasst auch die Hintergrundverarbeitung für den Datenbankserver und Verwaltungsaufgaben von Amazon DocumentDB. Wenn die Kapazität durch eines dieser Elemente eingeschränkt ist, skaliert DocumentDB Serverless nach oben. DocumentDB Serverless skaliert auch, wenn es Leistungsprobleme erkennt, die es auf diese Weise lösen kann. Sie können die Ressourcenauslastung und ihre Auswirkungen auf die serverlose DocumentDB-Skalierung überwachen, indem Sie die Verfahren unter verwenden. Überwachung von Amazon DocumentDB (serverlos)

Die Auslastung kann je nach Writer und Reader in Ihrem Cluster variieren. Der Writer verarbeitet Schreibvorgänge und führt alle Datenänderungen am Cluster-Volume durch. Leser können schreibgeschützte Anfragen verarbeiten.

Skalierung ist der Vorgang, der die serverlose DocumentDB-Kapazität für Ihre Datenbank erhöht oder verringert. Bei serverloser DocumentDB hat jeder Writer und Reader seinen eigenen aktuellen Kapazitätswert, der in DCUs gemessen wird. DocumentDB Serverless skaliert ein Schreib- oder Lesegerät auf eine höhere Kapazität, wenn die aktuelle Kapazität zu niedrig ist, um die Last zu

bewältigen. Der Writer oder Reader wird auf eine geringere Kapazität skaliert, wenn seine aktuelle Kapazität höher als erforderlich ist.

DocumentDB Serverless kann die Kapazität schrittweise erhöhen. Wenn Ihr Workload-Bedarf beginnt, die aktuelle Datenbankkapazität eines Writers oder Readers zu erreichen, erhöht DocumentDB Serverless die Anzahl der DCUs für diesen Writer oder Reader. DocumentDB Serverless skaliert die Kapazität in den Schritten, die erforderlich sind, um die optimale Leistung für die verbrauchten Ressourcen zu erzielen. Die Skalierung erfolgt in Schritten von nur 0,5 DCUs. Je größer die aktuelle Kapazität, desto größer ist das Skalierungsinkrement und desto schneller kann die Skalierung erfolgen.

Da die serverlose DocumentDB-Skalierung so häufig, granular und unterbrechungsfrei erfolgt, verursacht sie keine diskreten Ereignisse im. AWS-Managementkonsole Stattdessen können Sie die CloudWatch Amazon-Metriken wie `serverlessDatabaseCapacity` und messen und ihre Mindest-`DCUUtilization`, Höchst- und Durchschnittswerte im Zeitverlauf verfolgen. Weitere Informationen zur serverlosen Überwachung von DocumentDB finden Sie unter. Überwachung von Amazon DocumentDB (serverlos)

Das Hoch- oder Herunterskalieren kann folgende Ursachen haben:

- Speichernutzung
- CPU-Auslastung
- Netzwerkauslastung
- Speichernutzung

Sie können diese Ursachen der Skalierung up/down auf serverlosen DocumentDB-Instances überwachen. Weitere Informationen finden Sie unter Überwachung von Amazon DocumentDB (serverlos).

Sie können wählen, ob ein Reader gleichzeitig mit dem zugehörigen Writer oder unabhängig vom Writer skalieren soll. Dazu geben Sie die Hochstufungsstufe für diesen Reader an.

- Serverlose DocumentDB-Lesegeräte der Stufen 0 und 1 skalieren zur gleichen Zeit wie der Writer. Dieses Skalierungsverhalten macht Reader in den Prioritätsstufen 0 und 1 ideal verfügbar. Das liegt daran, dass sie immer auf die richtige Kapazität dimensioniert sind, um die Workload des Writers im Falle eines Failovers zu übernehmen.
- Reader der Hochstufungsstufen 2 bis 15 skalieren unabhängig vom Writer. Jeder Reader bleibt innerhalb der minimalen und maximalen DCU-Werte, die Sie für Ihren Cluster angegeben haben.

Wenn ein Reader unabhängig von der zugehörigen Writer-DB skaliert, kann er inaktiv werden und herunterskalieren, während der Writer weiterhin ein hohes Transaktionsvolumen verarbeitet. Er ist nach wie vor als Failover-Ziel verfügbar, wenn keine anderen Reader in niedrigeren Hochstufungsstufen zur Verfügung stehen. Wenn der Reader jedoch zum Writer hochgestuft wird, muss er möglicherweise hochskalieren, um die volle Workload des Writers zu bewältigen.

Einzelheiten zum Anzeigen und Ändern der Aufstiegsstufen für serverlose Instances finden Sie unter. Die Angebotsstufe für serverlose Lesegeräte anzeigen und ändern

Die serverlose DocumentDB-Skalierung kann erfolgen, während Datenbankverbindungen geöffnet sind, während Transaktionen ausgeführt werden usw. DocumentDB Serverless wartet nicht auf einen stillen Punkt, um mit der Skalierung zu beginnen. Die Skalierung unterbricht keine laufenden Datenbankoperationen.

Wenn Ihre Arbeitslast mehr Lesekapazität benötigt, als mit einem einzelnen Writer und einem einzelnen Reader verfügbar ist, können Sie dem Cluster mehrere serverlose DocumentDB-Lesegeräte hinzufügen. Jeder serverlose DocumentDB-Leser kann innerhalb des Bereichs der Mindest- und Höchstkapazitätswerte skaliert werden, den Sie für Ihren Cluster angegeben haben. Sie können den Reader-Endpunkt des Clusters verwenden, um schreibgeschützte Sitzungen an die Reader zu leiten und die Last des Writers zu reduzieren.

Ob DocumentDB Serverless eine Skalierung durchführt und wie schnell die Skalierung nach dem Start erfolgt, hängt auch von den minimalen und maximalen DCU-Einstellungen für den Cluster ab. Darüber hinaus hängt es davon ab, ob ein Reader so konfiguriert ist, dass er zusammen mit dem Writer oder unabhängig davon skaliert wird. Einzelheiten zur Skalierungskonfiguration finden Sie unter. Serverlose Skalierungskonfiguration von Amazon DocumentDB

Ruhezustand (0,5 DCUs)

Wenn serverlose Amazon DocumentDB-Writer oder -Lesegeräte inaktiv sind, unterstützen serverlose DocumentDB-Instances das Herunterskalieren auf einen Leerlaufzustand von 0,5 DCUs, wenn der Cluster auf 0,5 konfiguriert ist MinCapacity .

Im Ruhezustand verfügen die serverlosen DocumentDB-Instances nicht über genügend CPU-Rechenkapazität, um die meisten Produktionsworkloads zu unterstützen, sind aber bereit, schnell hochzuskalieren, um eine neue Arbeitslast zu unterstützen. In einem Zustand ohne Leerlauf benötigen serverlose DocumentDB-Instances in der Regel mindestens 1,0 — 2,5 DCUs. Wenn serverlose DocumentDB-Instances von einem Ruhezustand in einen Zustand ohne Leerlauf

hochskaliert werden, werden sie daher direkt auf 1,0 bis 2,5 DCUs hochskaliert (oder auf den Wert von, falls dieser niedriger ist). MaxCapacity

Um eine Herabskalierung auf 0,5 DCUs im Leerlauf zu unterstützen, werden Instance-Grenzwerte begrenzt, wenn sie so konfiguriert sind, dass sie kleiner oder gleich 1,0 DCUs sind. MinCapacity Weitere Informationen dazu, wie sich die MinCapacity Konfiguration auf die Grenzwerte auswirkt, finden Sie unter. Grenzwerte für serverlose Amazon DocumentDB-Instances

Anforderungen und Einschränkungen für DocumentDB serverless

Anforderungen an die Funktion

Verfügbarkeit in Regionen

Weitere Informationen darüber, AWS-Regionen wo der serverlose Instance-Typ von Amazon DocumentDB verfügbar ist, finden Sie in der Spalte Serverless unter. Unterstützte Instance-Klassen nach Region

Der folgende AWS CLI Befehl kann verwendet werden, um die genauen serverlosen DocumentDB-Instance-Optionen zu überprüfen, die in einer bestimmten Region angeboten werden:

```
aws docdb describe-orderable-db-instance-options \
        --region my_region \
        --db-instance-class db.serverless \
        --engine docdb
```

Verfügbarkeit der Engine-Version

DocumentDB serverless wird von den Amazon DocumentDB-Engine-Versionen 5.0 und 8.0 unterstützt. Es ist in den Engine-Versionen 3.6 oder 4.0 nicht verfügbar.

> **Note**
>
> DocumentDB Serverless wird nur in neueren Patch-Versionen von 5.0.0 unterstützt. Stellen Sie sicher, dass Ihr Cluster auf eine aktuelle Engine-Patch-Version aktualisiert ist. Weitere Informationen zur Wartung von Amazon DocumentDB-Clustern finden Sie unter Wartung von Amazon DocumentDB

Cluster-Konfiguration

Bevor Sie serverlose Amazon DocumentDB-Instances zu einem Amazon DocumentDB-Cluster hinzufügen, muss der Parameter auch für den Cluster festgelegt sein. `ServerlessV2ScalingConfiguration` Dies definiert den Kapazitätsbereich der serverlosen DocumentDB-Instances im Cluster. Weitere Informationen zur Skalierung der Konfiguration finden Sie unter. Serverlose Skalierungskonfiguration von Amazon DocumentDB

Einstellungen für den minimalen Skalierungskapazitätsbereich für bestimmte Amazon DocumentDB-Funktionen

Einige Funktionen von Amazon DocumentDB funktionieren mit serverloser Version von DocumentDB, können jedoch zu Problemen führen, wenn Ihr Kapazitätsbereich niedriger ist als für die Speicheranforderungen dieser Funktionen mit Ihrer spezifischen Arbeitslast erforderlich. In diesem Fall funktioniert Ihre Datenbank möglicherweise nicht so gut wie gewohnt oder es kann zu Fehlern wegen zu wenig Speicherplatz kommen.

Für den optimalen Betrieb der folgenden Funktionen ist eine Konfiguration mit einem höheren `MinCapacity` and/or `MaxCapacity` Wert erforderlich:

- Performance Insights
- Globale Cluster (gilt nur für den Primärcluster AWS-Region)
- Serverlose Instanzerstellung auf einem Cluster mit einem großen Datenvolumen

  Dazu gehört auch die serverlose Instanzerstellung im Rahmen einer Cluster-Wiederherstellung.

Empfehlungen zur Einstellung des geeigneten Kapazitätsbereichs (falls Sie diese Funktion verwenden) finden Sie unterAuswahl des Skalierungskapazitätsbereichs für einen serverlosen DocumentDB-Cluster. Informationen zur Fehlerbehebung, wenn für Ihre Datenbank aufgrund eines falsch konfigurierten Kapazitätsbereichs Fehler wegen zu wenig Speicherplatz auftreten, finden Sie unter Vermeiden von Fehlern aufgrund von zu wenig Speicher.

Einen Cluster erstellen, der Amazon DocumentDB serverlos verwendet

Einen serverlosen Amazon DocumentDB-Cluster erstellen

Mit Amazon DocumentDB serverless sind Ihre Cluster mit bereitgestellten Clustern austauschbar. Sie können Cluster verwenden, bei denen einige Instances serverlos arbeiten und einige Instances bereitgestellt werden.

Stellen Sie sicher, dass Ihre gewünschte Region und Engine-Version DocumentDB serverless unterstützen. Siehe Anforderungen und Einschränkungen für DocumentDB serverless.

Gehen Sie wie unter beschrieben vor, um einen Amazon DocumentDB-Cluster zu erstellen, in dem Sie serverlose Instances hinzufügen können. Erstellen eines Amazon DocumentDB-Clusters Der einzige Unterschied besteht darin, dass das `ServerlessV2ScalingConfiguration` Argument ebenfalls angegeben werden muss.

Das `ServerlessV2ScalingConfiguration` Argument gibt den Skalierungskapazitätsbereich Ihrer serverlosen DocumentDB-Instances an. Es besteht aus den minimalen und maximalen DCU (DocumentDB Capacity Unit) -Werten, die für alle serverlosen DocumentDB-Instances im Cluster gelten:

- Der `MinCapacity` Wert gibt die minimale Skalierungskapazität an.
- Der `MaxCapacity` Wert gibt die maximale Skalierungskapazität an.

Weitere Informationen zur Skalierung finden Sie unter Serverlose Skalierungskonfiguration von Amazon DocumentDB.

Using the AWS-Managementkonsole

Das folgende AWS-Managementkonsole Konfigurationsbeispiel zeigt, wie ein serverloser DocumentDB-Cluster erstellt wird.

1. Melden Sie sich bei der Amazon DocumentDB-Konsole an AWS-Managementkonsole und öffnen Sie sie.
2. Klicken Sie im Navigationsbereich auf Cluster.

> **Tip**
>
> Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

Die Cluster-Tabelle wird angezeigt.

3. Wählen Sie Erstellen aus.

   Die Seite „Amazon DocumentDB-Cluster erstellen“ wird angezeigt.

4. Wählen Sie auf der Seite „Amazon DocumentDB-Cluster erstellen“ im Abschnitt Cluster-Typ die Option Instance-based Cluster aus (dies ist die Standardoption).

5. Im Abschnitt Cluster-Konfiguration:

   a. Geben Sie als Cluster-ID einen eindeutigen Namen ein, z. **`myserverlesscluster`** B. Beachten Sie, dass die Konsole alle Clusternamen in Kleinbuchstaben umwandelt, unabhängig davon, wie sie eingegeben werden.

   b. Wählen Sie für die Engine-Version 5.0.0 aus (dies ist die Standardoption).

6. Wählen Sie im Abschnitt Cluster-Speicherkonfiguration Amazon DocumentDB Standard aus (dies ist die Standardoption).

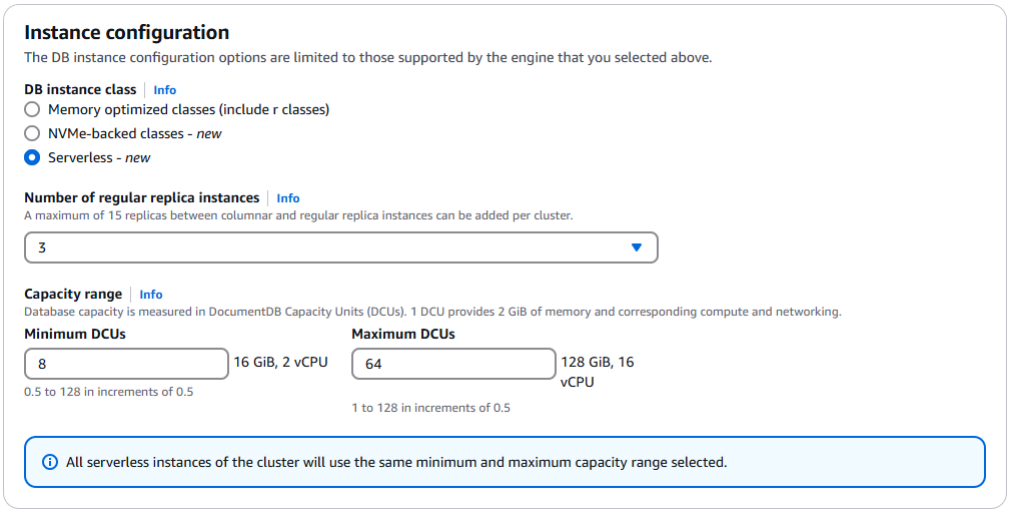
> **Note**
>
> Die andere Option in dieser Kategorie ist Amazon I/O-Optimized DocumentDB. Weitere Informationen zu beiden Optionen finden Sie unter Amazon DocumentDB-Cluster-Speicherkonfigurationen

7. Im Abschnitt Instanzkonfiguration:

   a. Wählen Sie die DB-Instance-Klasse Serverless aus.

   b. Wählen Sie für Anzahl regulärer Replikatinstanzen 3 aus (dies ist die Standardoption).

   c. Belassen Sie im Abschnitt Kapazitätsbereich die Standardwerte für Mindest-DCUs und Maximum-DCUs. Informationen zum Einstellen dieser Parameter finden Sie unter. Grenzwerte für serverlose Amazon DocumentDB-Instances



8. Belassen Sie im Abschnitt Konnektivität die Standardeinstellung Keine Verbindung zu einer EC2-Rechenressource herstellen.
9. Geben Sie im Abschnitt Authentifizierung einen Benutzernamen für den Hauptbenutzer ein und wählen Sie dann Selbstverwaltet aus. Geben Sie ein Passwort ein und bestätigen Sie es.

   Wenn Sie stattdessen Verwaltet in ausgewählt haben AWS Secrets Manager, finden Sie Passwortverwaltung mit Amazon DocumentDB und AWS Secrets Manager weitere Informationen unter.
10. Belassen Sie alle anderen Optionen als Standard und wählen Sie Cluster erstellen.

Using the AWS CLI

Ersetzen Sie im folgenden Beispiel jede *user input placeholder* durch Ihre eigenen Informationen oder Konfigurationsparameter.

Informationen zum Erstellen eines Clusters, das mit serverlosen DocumentDB-Instances kompatibel ist AWS CLI, mithilfe von finden Sie unter. Erstellen eines Clusters mit dem AWS CLI

Nehmen Sie die folgenden zusätzlichen Parameter in Ihren `create-db-cluster` Befehl auf:

```
--serverless-v2-scaling-configuration
     MinCapacity=minimum_capacity,MaxCapacity=maximum_capacity
```

Beispiel:

```
aws docdb create-db-cluster \
      --db-cluster-identifier sample-cluster \
      --engine docdb \
      --engine-version 5.0.0 \
      --serverless-v2-scaling-configuration MinCapacity=0.5,MaxCapacity=16 \
      --master-username user-name \
      --master-user-password password
```

Hinzufügen einer serverlosen Amazon DocumentDB-Instance

Um eine serverlose DocumentDB-Instance hinzuzufügen, gehen Sie wie unter beschrieben vor und achten Sie darauf[Hinzufügen einer Amazon DocumentDB DocumentDB-Instance zu einem Cluster](), db.serverless als Instance-Klasse anzugeben.

Hinzufügen einer serverlosen Instanz mit dem AWS-Managementkonsole.

Informationen zum Hinzufügen einer serverlosen DocumentDB-Instanz mithilfe der Konsole finden [Hinzufügen einer Amazon DocumentDB DocumentDB-Instance zu einem Cluster]() Sie unter Verwenden der Registerkarte. AWS-Managementkonsole

Hinzufügen einer serverlosen Instanz mithilfe der AWS CLI

Informationen zum Hinzufügen einer serverlosen DocumentDB-Instanz mithilfe von finden Sie unter Verwenden der AWS CLI Registerkarte [Hinzufügen einer Amazon DocumentDB DocumentDB-Instance zu einem Cluster]() und wählen Sie sie aus. AWS CLI

Verwenden Sie das folgende CLI-Argument für die Instanzklasse:

```
--db-instance-class db.serverless
```

Beispiel:

```
aws docdb create-db-instance \
      --db-cluster-identifier sample-cluster \
      --db-instance-identifier sample-instance \
      --db-instance-class db.serverless \
      --engine docdb
```

Migration zu Amazon DocumentDB serverlos

Topics

- Migration vorhandener DocumentDB-Cluster zu serverlosen Clustern
- Migration von MongoDB zu DocumentDB serverless

Migration vorhandener DocumentDB-Cluster zu serverlosen Clustern

Aktualisierung der Engine-Version eines Clusters

Wenn auf Ihrem bereitgestellten Cluster eine niedrigere Engine-Version ausgeführt wird, die DocumentDB serverless nicht unterstützt, müssen Sie den Cluster zunächst auf eine unterstützte Engine-Version aktualisieren. Weitere Informationen finden Sie unter Direktes Upgrade der Hauptversion von Amazon DocumentDB.

Migrieren Sie einen bereitgestellten Cluster zu DocumentDB serverless

Gehen Sie folgendermaßen vor, um einen bereitgestellten Cluster auf die serverlose Verwendung von DocumentDB umzustellen:

1. Prüfen Sie, ob die Engine-Version des bereitgestellten Clusters aktualisiert werden muss, um mit DocumentDB serverless verwendet zu werden. Siehe Anforderungen und Einschränkungen für DocumentDB serverless.

   > **Note**
   >
   > Wenn auf dem bereitgestellten Cluster eine Engine-Version ausgeführt wird, die für DocumentDB serverless nicht verfügbar ist, aktualisieren Sie die Engine-Version des Clusters. Siehe Direktes Upgrade der Hauptversion von Amazon DocumentDB.

2. Konfigurieren Sie die Skalierungskonfiguration für den Cluster. Einzelheiten zur Auswahl der Skalierungskonfiguration finden Sie unter. Auswahl des Skalierungskapazitätsbereichs für einen serverlosen DocumentDB-Cluster Informationen zum Ändern der Skalierungskonfiguration für einen Cluster finden Sie unterDie Konfiguration des Skalierungskapazitätsbereichs eines Clusters anzeigen und ändern.

3. Konfigurieren Sie alle anderen Clustereigenschaften, um die Anforderungen und Einschränkungen von DocumentDB serverless zu berücksichtigen. Anforderungen und Einschränkungen für DocumentDB serverless
4. Fügen Sie dem Cluster eine oder mehrere serverlose DocumentDB-Instanzen hinzu. Folgen Sie dem Verfahren unter Hinzufügen einer serverlosen Amazon DocumentDB-Instance.

> **Note**
>
> In einigen Fällen verfügen Sie möglicherweise bereits über eine oder mehrere bereitgestellte Reader-Instanzen im Cluster. In diesem Fall können Sie einen der Reader in eine serverlose DocumentDB-Instanz konvertieren, anstatt eine neue Instanz zu erstellen. Eine Schritt-für-Schritt-Anleitung hierzu finden Sie unter Die Klasse einer Instance ändern.

5. (Optional) Führen Sie einen Failover-Vorgang durch, um eine serverlose DocumentDB-Instanz zum Cluster-Writer zu machen. Siehe Amazon DocumentDB-Failover.
6. (Optional) Konvertieren Sie alle verbleibenden bereitgestellten Amazon DocumentDB-Instances in serverlose DocumentDB-Instances (sieheDie Klasse einer Instance ändern) oder entfernen Sie sie aus dem Cluster (siehe). Löschen einer Amazon DocumentDB DocumentDB-Instance

Using the AWS-Managementkonsole

Das folgende AWS-Managementkonsole Konfigurationsbeispiel zeigt den Migrationsprozess unter Verwendung eines von Amazon DocumentDB bereitgestellten Clusters, auf dem Amazon DocumentDB 5.0.0 ausgeführt wird. Für den Start der serverlosen Nutzung von DocumentDB ist kein Upgrade der Engine-Version erforderlich. Der Cluster ist benannt und beginnt mit drei bereitgestellten Instances mit den Namen`sample`, und. `sample` `sample2` `sample3` In diesem Beispiel werden diese drei Instanzen durch drei serverlose Instanzen ersetzt.

1. Melden Sie sich bei der Amazon DocumentDB-Konsole an AWS-Managementkonsole und öffnen Sie sie.
2. Klicken Sie im Navigationsbereich auf Cluster.

> **Tip**
>
> Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

Die Cluster-Tabelle wird angezeigt.

3. Wählen Sie in der Cluster-Tabelle das Kontrollkästchen des Clusters aus, zu dem Sie eine serverlose Instance hinzufügen möchten.
4. Wählen Sie Actions (Aktionen) und dann Add instances (Instances hinzufügen) aus.
5. Wählen Sie im Dialogfeld Instanzen hinzufügen im Abschnitt DB-Instance-Klasse für jede neue serverlose Instance, die Sie erstellen möchten, die Option Serverless aus.
6. Legen Sie für serverlose Kapazitätseinstellungen die Skalierungskonfiguration auf der Grundlage der Kapazitätsbeschreibung im Dialogfeld fest.
7. (Optional) Um eine weitere Instanz hinzuzufügen, wählen Sie Instanz hinzufügen. Fügen Sie weitere Instanzen hinzu, bis Sie die gewünschte Anzahl neuer Instanzen erreicht haben.

   In diesem Beispiel werden drei neue serverlose Instanzen erstellt.
8. Wählen Sie Erstellen aus.

   Es dauert einige Minuten, bis eine Instance erstellt ist. Sie können die Konsole oder die verwenden AWS CLI , um den Status der Instanz einzusehen. Weitere Informationen finden Sie unter Überwachung des Status eines Amazon DocumentDB-Clusters.
9. Kehren Sie zur Cluster-Tabelle zurück und aktivieren Sie die Kontrollkästchen aller drei ursprünglich bereitgestellten Instanzen.
10. Wählen Sie Aktionen und anschließend Löschen aus.

Während des Löschvorgangs wird automatisch ein Failover durchgeführt, um eine der verbleibenden Instanzen zum Writer hochzustufen. Nach einigen Minuten ist der Löschvorgang abgeschlossen. Der bestehende Cluster enthält jetzt drei serverlose DocumentDB-Instanzen (wie in der Spalte Größe definiert).

Using the AWS CLI

Das folgende AWS CLI Konfigurationsbeispiel zeigt den Migrationsprozess unter Verwendung eines von Amazon DocumentDB bereitgestellten Clusters, auf dem Amazon DocumentDB 5.0.0 ausgeführt wird. Für die Nutzung von DocumentDB serverless ist kein Upgrade der Engine-Version erforderlich. Der Cluster ist benannt und beginnt mit drei bereitgestellten Instances mit den Namen`sample`, und. `sample` `sample2` `sample3` In diesem Beispiel werden diese drei Instanzen durch drei serverlose Instanzen ersetzt. Der Cluster ist benannt `sample-cluster` und beginnt mit zwei bereitgestellten Instanzen mit dem Namen `sample-provisioned-instance-1` und`sample-provisioned-instance-2`, einer Writer-Instanz und einer Reader-Instanz.

Ersetzen Sie im folgenden Beispiel jede *`user input placeholder`* durch Ihre eigenen Informationen oder Konfigurationsparameter.

Verwenden Sie den `aws docdb describe-db-clusters` Vorgang, um den Status eines Clusters zu ermitteln. Der folgende Code ermittelt den Status des Clusters `sample-cluster` und gibt die Ergebnisse in einer Tabelle aus:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].DBClusterMembers' \
    --output table
```

```
-------------------------------------------------------------------------------------------------
|                                          DescribeDBClusters
                  |
+---------------------------------+---------------------------------
+------------------+----------------+
|  DBClusterParameterGroupStatus |      DBInstanceIdentifier       | IsClusterWriter
  | PromotionTier  |
+---------------------------------+---------------------------------
+------------------+----------------+
|  in-sync                       |  sample-provisioned-instance-2  |  False
  |  1             |
|  in-sync                       |  sample-provisioned-instance-1  |  True
  |  1             |
+---------------------------------+---------------------------------
+------------------+----------------+
```

Fügen Sie die Skalierungskonfiguration für den Cluster hinzu:

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --serverless-v2-scaling-configuration MinCapacity=0.5,MaxCapacity=16
```

Fügen Sie die serverlosen Instanzen hinzu. In diesem Beispiel werden neue serverlose Instanzen benannt `sample-serverless-instance-1` und hinzugefügt`sample-serverless-instance-2`:

```
aws docdb create-db-instance \
    --db-cluster-identifier sample-cluster \
    --db-instance-identifier sample-serverless-instance-1 \
    --db-instance-class db.serverless \
    --engine docdb

aws docdb create-db-instance \
    --db-cluster-identifier sample-cluster \
    --db-instance-identifier sample-serverless-instance-2 \
    --db-instance-class db.serverless \
    --engine docdb
```

Geben Sie Folgendes ein, um zu warten, bis die serverlosen Instanzen verfügbar sind, bevor Sie fortfahren:

```
aws docdb wait db-instance-available \
    --db-instance-identifier sample-serverless-instance-1

aws docdb wait db-instance-available \
    --db-instance-identifier sample-serverless-instance-2
```

Führen Sie einen Failover durch, um die neue `sample-serverless-instance-1` Instanz zum Cluster-Writer zu machen:

```
aws docdb failover-db-cluster \
    --db-cluster-identifier sample-cluster \
    --target-db-instance-identifier sample-serverless-instance-1
```

Das Failover dauert einige Sekunden. Danach wird sample-serverless-instance-1 zum Cluster-Writer. Überprüfen Sie dies mit der folgenden Eingabe:

```
aws docdb describe-db-clusters \
```

```
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].DBClusterMembers' \
    --output table
```

```
-------------------------------------------------------------------------------------------------
|                                          DescribeDBClusters
                  |
+---------------------------------+---------------------------------
+------------------+----------------+
|  DBClusterParameterGroupStatus |      DBInstanceIdentifier       | IsClusterWriter
  | PromotionTier  |
+---------------------------------+---------------------------------
+------------------+----------------+
|  in-sync                       |  sample-provisioned-instance-2  |  False
  |  1             |
|  in-sync                       |  sample-provisioned-instance-1  |  False
  |  1             |
|  in-sync                       |  sample-serverless-instance-2   |  False
  |  1             |
|  in-sync                       |  sample-serverless-instance-1   |  True
  |  1             |
+---------------------------------+---------------------------------
+------------------+----------------+
```

Löschen Sie abschließend die ursprünglich bereitgestellten Instanzen:

```
aws docdb delete-db-instance \
    --db-instance-identifier sample-provisioned-instance-1

aws docdb delete-db-instance \
    --db-instance-identifier sample-provisioned-instance-2
```

Migration von MongoDB zu DocumentDB serverless

Sie können Ihre MongoDB-Datenbanken auf DocumentDB serverless migrieren, genau wie bei bereitgestelltem Amazon DocumentDB. Weitere Informationen finden Sie unter Migration und Aktualisierung von Amazon DocumentDB.

Verwaltung von Amazon DocumentDB serverlos

Die Konfiguration des Skalierungskapazitätsbereichs eines Clusters anzeigen und ändern

Das `ServerlessV2ScalingConfiguration` Argument gibt den Skalierungskapazitätsbereich Ihrer serverlosen DocumentDB-Instances an. Es besteht aus den minimalen und maximalen DCU (DocumentDB Capacity Unit) -Werten, die für alle serverlosen DocumentDB-Instances im Cluster gelten.

- **`MinCapacity`**— Die minimale Skalierungskapazität aller serverlosen DocumentDB-Instances im Cluster.
- **`MaxCapacity`**— Die maximale Skalierungskapazität aller serverlosen DocumentDB-Instanzen im Cluster.

> **Note**
>
> Für die folgenden Änderungen der Skalierungskonfiguration ist ein Neustart der Instanz erforderlich, um die neuen `MinCapacity` UND-Werte widerzuspiegeln: `MaxCapacity`
>
> - Alle Änderungen des `MaxCapacity` Werts
> - Änderung des `MinCapacity` Werts von einem höheren Wert auf 1,0 oder niedriger
> - Änderung des `MinCapacity` Werts von einem niedrigeren Wert auf einen Wert über 1,0

Weitere Informationen zur Skalierungskonfiguration und zur Auswahl geeigneter Skalierungskapazitätsgrenzen finden Sie unterServerlose Skalierungskonfiguration von Amazon DocumentDB.

Using the AWS-Managementkonsole

Das folgende AWS-Managementkonsole Konfigurationsbeispiel zeigt, wie die Skalierungskonfigurationseinstellungen eines serverlosen DocumentDB-Clusters angezeigt und bearbeitet werden.

1. Melden Sie sich bei der Amazon DocumentDB-Konsole an AWS-Managementkonsole und öffnen Sie sie.

2. Klicken Sie im Navigationsbereich auf Cluster.

> **Tip**
>
> Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

Die Cluster-Tabelle wird angezeigt.

3. Wählen Sie in der Cluster-Tabelle das Kontrollkästchen des Clusters aus, für den Sie die Skalierungskapazität ändern möchten.
4. Wählen Sie Actions (Aktionen) und dann Modify (Ändern) aus.

   Das Dialogfeld „Cluster ändern“ wird angezeigt.
5. Suchen Sie den Abschnitt „Serverlose Kapazitätseinstellung“ und legen Sie die Skalierungskonfiguration (Kapazitätsbereich) auf der Grundlage der Kapazitätsbeschreibung im Dialogfeld fest.

   Weitere Informationen zur Skalierung und zu Kapazitätsbereichen finden Sie unterServerlose Skalierungskonfiguration von Amazon DocumentDB.
6. Klicken Sie auf Weiter.
7. Wählen Sie für Scheduling of modifications (Einplanung von Änderungen) die Option Apply immediately (Sofort anwenden) aus.
8. Wählen Sie Cluster bearbeiten aus.
9. Sobald die Änderung abgeschlossen ist, sollte jede serverlose Instanz neu gestartet werden. Um die Nichtverfügbarkeit des Writers zu minimieren, führen Sie die folgende Abfolge von Vorgängen aus:

   a. Starten Sie jede serverlose Reader-Instanz neu.

      i. Wählen Sie die Reader-Instanz aus, wählen Sie Aktionen und dann Reboot.

      ii. Warten Sie, bis der Instanzstatus wieder verfügbar ist.

   b. Führen Sie ein Failover zu einer neu gestarteten serverlosen Instance durch.

      i. Wählen Sie den Cluster aus, wählen Sie Aktionen und dann Failover.

ii. Warten Sie, bis der Failover-Vorgang abgeschlossen ist.

c. Starten Sie die verbleibende serverlose Instanz neu.

i. Wählen Sie die verbleibende Instanz aus, wählen Sie Aktionen und dann Neustart.

ii. Warten Sie, bis der Instanzstatus wieder verfügbar ist.

Using the AWS CLI

Das folgende AWS CLI Konfigurationsbeispiel zeigt die aktuelle Skalierungskonfiguration.

Ersetzen Sie im folgenden Beispiel jedes *user input placeholder* durch Ihre eigenen Informationen oder Konfigurationsparameter.

Die aktuelle Skalierungskonfiguration des Clusters kann mit dem `describe-db-clusters` AWS CLI folgenden Befehl angezeigt werden:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[0].ServerlessV2ScalingConfiguration'
```

Die Ausgabe dieses Befehls ist wie folgt:

```
{
    "MinCapacity": 0.5,
    "MaxCapacity": 16.0
}
```

Die Skalierungskonfiguration des Clusters kann mit dem `modify-db-cluster` folgenden Befehl geändert werden:

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --serverless-v2-scaling-configuration MinCapacity=0.5,MaxCapacity=32
```

Sobald der Vorgang abgeschlossen ist, sollte jede serverlose Instanz neu gestartet werden. Um die Nichtverfügbarkeit des Writers zu minimieren, können wir die folgende Abfolge von Vorgängen ausführen:

```
aws docdb reboot-db-instance \
    --db-instance-identifier sample-serverless-instance-reader
```

```
aws docdb wait db-instance-available \
    --db-instance-identifier sample-serverless-instance-reader

aws docdb failover-db-cluster \
   --db-cluster-identifier sample-cluster \
   --target-db-instance-identifier sample-serverless-instance-reader

aws docdb reboot-db-instance \
    --db-instance-identifier sample-serverless-instance-writer

aws docdb wait db-instance-available \
    --db-instance-identifier sample-serverless-instance-writer

aws docdb failover-db-cluster \
   --db-cluster-identifier sample-cluster \
   --target-db-instance-identifier sample-serverless-instance-writer
```

Die Angebotsstufe für serverlose Lesegeräte anzeigen und ändern

Achten Sie bei Clustern, die mehrere serverlose DocumentDB-Instances oder eine Mischung aus bereitgestellten und serverlosen Instances enthalten, auf die Einstellung der Aufstiegsstufe für jede serverlose Instance. Diese Einstellung steuert mehr Verhalten für serverlose Instances als für bereitgestellte Instances.

Bei bereitgestellten Instances bestimmt die Wahl der Stufen 0—15 nur die Reihenfolge, in der Amazon DocumentDB auswählt, welche Reader-Instance während eines Failover-Vorgangs an den Writer heraufgestuft werden soll. Bei serverlosen Instances bestimmt die Tier-Nummer jedoch auch, ob die Instance entsprechend der Kapazität der Writer-Instance hochskaliert wird, oder ob sie unabhängig von ihrer eigenen Arbeitslast skaliert wird. Serverlose Reader-Instances der Stufe 0 oder 1 werden auf einer Mindestkapazität gehalten, die mindestens so hoch ist wie die Writer-Instance. Auf diese Weise sind sie bereit, im Falle eines Failovers die Writer-Instanz zu ersetzen. Wenn es sich bei der Writer-Instance um eine bereitgestellte Instance handelt, schätzt Amazon DocumentDB die entsprechende serverlose DocumentDB-Kapazität. Diese Schätzung wird als Mindestkapazität für die serverlose Reader-Instance verwendet.

Für serverlose DocumentDB-Leser-Instances der Stufen 2—15 gelten nicht dieselben Beschränkungen in Bezug auf ihre Mindestkapazität. Wenn sie inaktiv sind, können sie auf den Mindestwert der DocumentDB-Kapazitätseinheit (DCU) herunterskaliert werden, der im Kapazitätsbereich des Clusters angegeben ist.

Using the AWS-Managementkonsole

Das folgende AWS-Managementkonsole Konfigurationsbeispiel zeigt, wie die Einstellungen für die Werbestufe eines serverlosen DocumentDB-Instance-Readers angezeigt und geändert werden.

1. Melden Sie sich bei der Amazon DocumentDB-Konsole an AWS-Managementkonsole und öffnen Sie sie.

   Die Promotion-Stufen jeder Instance werden in der Spalte Promotion-Stufe in der AWS-Managementkonsole angezeigt.

2. Klicken Sie im Navigationsbereich auf Cluster.

   > **Tip**
   >
   > Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

   Die Cluster-Tabelle wird angezeigt.

3. Wählen Sie in der Cluster-Tabelle das Kontrollkästchen der Instance aus, für die Sie die Promotion-Stufe ändern möchten.
4. Wählen Sie Actions (Aktionen) und dann Modify (Ändern) aus.

   Das Dialogfeld „Instanz ändern“ wird angezeigt.

5. Suchen Sie den Abschnitt Failover und stellen Sie die Promotion-Stufe auf die gewünschte Stufe ein.
6. Klicken Sie auf Weiter.
7. Wählen Sie für Scheduling of modifications (Einplanung von Änderungen) die Option Apply immediately (Sofort anwenden) aus.
8. Wählen Sie Modify instance (Instance ändern).

Using the AWS CLI

Das folgende AWS CLI Konfigurationsbeispiel zeigt die aktuelle Skalierungskonfiguration.

Ersetzen Sie im folgenden Beispiel jedes *user input placeholder* durch Ihre eigenen Informationen oder Konfigurationsparameter.

Die Promotion-Stufen aller Instances in einem Cluster können mit dem `describe-db-clusters` AWS CLI folgenden Befehl angezeigt werden:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[0].DBClusterMembers' \
    --output table
```

Die Ausgabe dieses Befehls ist wie folgt:

```
----------------------------------------------------------------------------------------------
|                                          DescribeDBClusters
                  |
+---------------------------------+---------------------------------
+------------------+----------------+
|  DBClusterParameterGroupStatus |      DBInstanceIdentifier       | IsClusterWriter
  | PromotionTier  |
+---------------------------------+---------------------------------
+------------------+----------------+
|  in-sync                       |  sample-serverless-instance-2   |  False
  |  1             |
|  in-sync                       |  sample-serverless-instance-1   |  True
  |  1             |
+---------------------------------+---------------------------------
+------------------+----------------+
```

Die Promotion-Stufe einer bestimmten Instanz kann mit dem `modify-db-instance` folgenden Befehl geändert werden:

```
aws docdb modify-db-instance \
    --db-instance-identifier sample-serverless-instance-2 \
    --promotion-tier 3
```

Grenzwerte für serverlose Amazon DocumentDB-Instances

Für serverlose DocumentDB-Instances hängen die folgenden Grenzwerte pro Instance von der aktuellen Skalierungskapazität der Instance ab:

- Instanzspeicher (GiB)
- Verbindungen (alle)
- Cursor-Limit
- Transaktionen öffnen
- Verbindungen (aktiv)

In den folgenden Tabellen wird beschrieben, wie die Grenzwerte für serverlose Instances pro Instanz im Verhältnis zur aktuellen Skalierungskapazität der Instance skaliert werden.

Beachten Sie, dass die Grenzwerte unterschiedlich sind, je nachdem, ob die `MinCapacity` Skalierungskonfiguration des Clusters auf mehr als zwei (2) festgelegt wurde. Wenn der Wert auf kleiner oder gleich 2 gesetzt `MinCapacity` ist, sind die Grenzwerte für Cursorlimit, offene Transaktionen und Verbindungen (aktiv) auf einen niedrigeren Höchstwert begrenzt. (Der technische Grund für die Obergrenze besteht darin, die Skalierung auf einen „inaktiven“ Zustand zu unterstützen. Weitere Informationen finden Sie unter Ruhezustand (0,5 DCUs).

Limits für db.serverless-Instances, wenn ≤1 MinCapacity

Aktuelle Kapazität (DCU)	Instanzspeicher (GiB)	Verbindungen (alle)	Cursor-Limit	Transaktionen öffnen	Verbindungen (aktiv)
0.5	1	250	6	12	39
1	2	500	12	24	79
2	4	1000	24	48	173
4	8	2000	48	96	416
8	16	4000	96	192	1071
16	32	8000	132	264	1550
32	64	16000	132	264	1550
64	128	32000	132	264	1550
128	256	60000	132	264	1550

Aktuelle Kapazität (DCU)	Instanzspeicher (GiB)	Verbindungen (alle)	Cursor-Limit	Transaktionen öffnen	Verbindungen (aktiv)
256	512	60000	132	264	1550

Limits für db.serverless-Instanzen, wenn >1 MinCapacity

Aktuelle Kapazität (DCU)	Instanzspeicher (GiB)	Verbindungen (alle)	Cursor-Limit	Transaktionen öffnen	Verbindungen (aktiv)
1.5	3	750	18	36	124
2	4	1000	24	48	173
4	8	2000	48	96	416
8	16	4000	96	192	1071
16	32	8000	192	384	2709
32	64	16000	384	768	4500
64	128	32000	768	1536	4500
128	256	60000	1536	3072	4500
256	512	60000	3072	6144	4500

Mithilfe der folgenden CloudWatch Metriken können Sie die Grenzwerte pro Instanz überwachen und Warnmeldungen einleiten. Weitere Informationen zu Amazon CloudWatch DocumentDB-Metriken finden Sie unterÜberwachung von Amazon DocumentDB mit CloudWatch.

Ressource	CloudWatch Metrik einschränken	CloudWatch Nutzungsmetrik (maximal 1 Minute)	CloudWatch Metrik zur Nutzung
Instanzspeicher	-	-	FreeableMemory
Verbindungen (alle)	DatabaseConnectionsLimit	DatabaseConnectionsMax	DatabaseConnections
Cursor	DatabaseCursorsLimit	DatabaseCursorsMax	DatabaseCursors
Transaktionen	TransactionsOpenLimit	TransactionsOpenMax	TransactionsOpen

Serverlose Skalierungskonfiguration von Amazon DocumentDB

Topics

- Auswahl des Skalierungskapazitätsbereichs für einen serverlosen DocumentDB-Cluster
- Auswahl der MinCapacity Einstellung für einen serverlosen DocumentDB-Cluster
- Auswahl der MaxCapacity Einstellung für einen serverlosen DocumentDB-Cluster
- Vermeiden von Fehlern aufgrund von zu wenig Speicher
- Warum wird meine serverlose Instance nicht herunterskaliert?

Auswahl des Skalierungskapazitätsbereichs für einen serverlosen DocumentDB-Cluster

Bevor serverlose DocumentDB-Instances zu einem Amazon DocumentDB-Cluster hinzugefügt werden, muss der Parameter auch für den Cluster festgelegt sein. `ServerlessV2ScalingConfiguration`

Der `ServerlessV2ScalingConfiguration` Parameter besteht aus zwei Werten, die den Kapazitätsbereich für die serverlose Skalierung jeder serverlosen Instance im Cluster definieren:

- **`MinCapacity`**— Die minimale Skalierungskapazität aller serverlosen DocumentDB-Instanzen im Cluster.

- **`MaxCapacity`**— Die maximale Skalierungskapazität aller serverlosen DocumentDB-Instances im Cluster.

Auswahl der **`MinCapacity`** Einstellung für einen serverlosen DocumentDB-Cluster

Es ist verlockend, immer 0,5 für zu wählen. `MinCapacity` Mit diesem Wert kann die Instance auf die kleinste Kapazität herunterskaliert werden, wenn sie vollständig inaktiv ist, während sie aktiv bleibt. Je nachdem, wie Sie diesen Cluster verwenden und wie die anderen Einstellungen konfiguriert sind, ist jedoch eine andere Mindestkapazität ggf. am effektivsten. Berücksichtigen Sie bei der Auswahl der Mindestkapazitätseinstellung die folgenden Faktoren:

- Die Skalierungsrate für eine serverlose DocumentDB-Instance hängt von ihrer aktuellen Kapazität ab. Je höher die aktuelle Kapazität, desto schneller kann sie hochskalieren. Wenn die Instance schnell auf eine sehr hohe Kapazität hochskaliert werden muss, sollten Sie erwägen, die Mindestkapazität auf einen Wert festzulegen, bei dem die Skalierungsrate Ihren Anforderungen entspricht.
- Wenn Sie in der Regel die Instance-Klasse Ihrer Instances in Erwartung einer besonders hohen oder niedrigen Arbeitslast ändern, können Sie diese Erfahrung nutzen, um eine grobe Schätzung des äquivalenten serverlosen Kapazitätsbereichs von DocumentDB vorzunehmen. Informationen zur Bestimmung der Speichergröße eines bereitgestellten Amazon DocumentDB-Instance-Typs finden Sie unter. Kontingente für Instanzen

  Nehmen wir beispielsweise an, Sie verwenden die `db.r6g.xlarge` Instance-Klasse, wenn Ihr Cluster eine geringe Arbeitslast hat. Diese Instanzklasse hat 32 GiB Speicher. Sie können also einen Wert `MinCapacity` von 16 angeben, um eine serverlose Instanz einzurichten, die auf ungefähr dieselbe Kapazität herunterskaliert werden kann. Das liegt daran, dass jede DCU ungefähr 2 GiB Arbeitsspeicher entspricht. Sie könnten einen etwas niedrigeren Wert angeben, um die Instance weiter herunterskalieren zu lassen, falls Ihre `db.r6g.xlarge` Instance manchmal nicht ausgelastet war.
- Wenn Ihre Anwendung am effizientesten arbeitet, wenn die Instances eine bestimmte Datenmenge im Puffer-Cache haben, sollten Sie erwägen, eine DCU-Mindesteinstellung anzugeben, bei der der Speicher groß genug ist, um die häufig aufgerufenen Daten aufzunehmen. Andernfalls werden einige Daten aus dem Puffer-Cache entfernt, wenn die serverlosen Instances auf eine niedrigere Speichergröße herunterskaliert werden. Wenn die Instanzen dann wieder hochskaliert werden, werden die Informationen im Laufe der Zeit wieder in den Puffer-Cache eingelesen. Wenn

die Menge I/O , die zurück in den Puffercache gebracht werden muss, erheblich ist, könnte es effektiver sein, einen höheren DCU-Mindestwert zu wählen. Weitere Informationen finden Sie unter Dimensionierung der Instanz.

- Wenn Ihre serverlosen DocumentDB-Instances die meiste Zeit mit einer bestimmten Kapazität ausgeführt werden, sollten Sie erwägen, eine Mindestkapazitätseinstellung anzugeben, die unter dem Basiswert liegt, aber nicht zu viel niedriger ist. Serverlose Instances können am effektivsten abschätzen, um wie viel und wie schnell eine Skalierung erfolgen muss, wenn die aktuelle Kapazität nicht drastisch unter der erforderlichen Kapazität liegt.
- Wenn Ihr bereitgestellter Workload Speicheranforderungen hat, die für kleine Instance-Klassen wie T3 oder T4g zu hoch sind, wählen Sie eine DCU-Mindesteinstellung, die Arbeitsspeicher bereitstellt, der mit einer R5- oder R6g-Instance vergleichbar ist.
- Verwenden Sie `MinCapacity` für die angegebenen Funktionen mindestens das folgende Minimum (Änderungen vorbehalten):
  - Einblicke in die Leistung — 2 DCUs
  - Globale Cluster — 8 DCUs (gilt nur für den primären) AWS-Region
- In Amazon DocumentDB erfolgt die Replikation auf der Speicherebene, sodass sich die Lesekapazität nicht direkt auf die Replikation auswirkt. Stellen Sie jedoch bei serverlosen DocumentDB-Leser-Instances, die unabhängig skaliert werden, sicher, dass die Mindestkapazität ausreicht, um Arbeitslasten in schreibintensiven Zeiten zu bewältigen, um Abfragelatenzen zu vermeiden. Wenn bei Reader-Instances der Stufen 2—15 Leistungsprobleme auftreten, sollten Sie erwägen, die Mindestkapazität des Clusters zu erhöhen. Einzelheiten dazu, wie Sie ändern können, ob Reader-Instances zusammen mit dem Writer oder unabhängig skalieren, finden Sie unterDie Angebotsstufe für serverlose Lesegeräte anzeigen und ändern.

  Wenn Sie einen Cluster mit serverlosen DocumentDB-Leser-Instanzen haben, skalieren die Reader nicht zusammen mit der Writer-Instanz, wenn die Aufstiegsstufe der Leser nicht 0 oder 1 ist. In diesem Fall kann das Festlegen einer geringen Mindestkapazität zu einer übermäßigen Replikationsverzögerung führen. Das liegt daran, dass die Reader möglicherweise nicht genug Kapazität haben, um Änderungen vom Writer zu übernehmen, wenn die Datenbank ausgelastet ist. Stellen Sie die Mindestkapazität auf einen Wert ein, der einer mit der Writer-Instanz vergleichbaren Menge an Arbeitsspeicher und CPU entspricht.
- Die Zeit, die eine serverlose DocumentDB-Instance benötigt, um von ihrer Mindestkapazität auf ihre maximale Kapazität zu skalieren, hängt von der Differenz zwischen ihren minimalen und maximalen DCU-Werten ab. Wenn die aktuelle Kapazität der Instanz groß ist, skaliert DocumentDB Serverless in größeren Schritten nach oben, als wenn die Instance mit einer geringen Kapazität

gestartet wird. Wenn Sie also eine relativ große maximale Kapazität angeben und die Instance die meiste Zeit in der Nähe dieser Kapazität verbringt, sollten Sie erwägen, die Mindest-DCU-Einstellung zu erhöhen. Auf diese Weise kann eine Instance im Leerlauf schneller wieder auf die maximale Kapazität hochskaliert werden.

- Bestimmte Instance-Limits werden von der aktuellen Kapazität der serverlosen Instance bestimmt, z. B. Verbindungslimit, Cursorlimit und Limit für offene Transaktionen. Wenn die aktuelle Kapazität der Instance gering ist, sind auch die Limits entsprechend gering. Wenn diese Grenzwerte ein Problem darstellen, wenn Ihre serverlose Instance auf ihren `MinCapacity` Wert herunterskaliert wird, sollten Sie eine Erhöhung `MinCapacity` auf einen höheren Wert in Betracht ziehen. Weitere Informationen finden Sie unter Grenzwerte für serverlose Amazon DocumentDB-Instances.
- Darüber hinaus sind bestimmte Instance-Limits auf einen niedrigeren Maximalwert begrenzt, wenn der auf weniger als oder gleich 1,0 DCUs festgelegt `MinCapacity` ist, wie z. B. das Limit für aktive Verbindungen, das Cursorlimit und das Limit für offene Transaktionen. Wenn diese Obergrenzen für Ihre Arbeitslast nicht ausreichen, verwenden Sie einen `MinCapacity` Wert von mindestens 1,5 DCUs. Weitere Informationen finden Sie unter Grenzwerte für serverlose Amazon DocumentDB-Instances.

Anweisungen zum Ändern der Skalierungskonfiguration eines Clusters finden Sie unter. Verwaltung von Amazon DocumentDB serverlos

Auswahl der **`MaxCapacity`** Einstellung für einen serverlosen DocumentDB-Cluster

Es ist verlockend, immer einen hohen Wert für die maximale serverlose DocumentDB-Kapazitätseinstellung zu wählen. Eine große maximale Kapazität ermöglicht es der Instance, am stärksten zu skalieren, wenn sie eine intensive Arbeitslast ausführt. Mit einem niedrigen Wert entfällt die Möglichkeit unerwarteter Gebühren. Je nachdem, wie Sie diesen Cluster verwenden und die anderen Einstellungen konfigurieren, kann der effektivste Wert höher oder niedriger sein, als Sie ursprünglich dachten. Berücksichtigen Sie bei der Auswahl der maximalen Kapazitätseinstellung die folgenden Faktoren:

- Die maximale Kapazität muss mindestens so hoch sein wie die Mindestkapazität. Sie können die minimale und maximale Kapazität auf den gleichen Wert festlegen. In diesem Fall skaliert sich die Kapazität jedoch niemals hoch oder herunter. Daher ist die Verwendung identischer Werte für die minimale und maximale Kapazität außerhalb von Testsituationen nicht geeignet.
- Die maximale Kapazität muss mindestens 1,0 DCUs und höchstens 256 DCUs betragen.

- Überwachen Sie die Skalierung und den Ressourcenverbrauch Ihrer serverlosen Instances. Wenn Ihre serverlose Instance häufig bis zur maximalen Kapazität skaliert wird und dabei an Ressourcenbeschränkungen stößt (z. B. wenn die `DCUUtilization` Metrik bei 100,0 liegt), wählen Sie einen höheren Wert. `MaxCapacity` Weitere Informationen finden Sie unter Überwachung von Amazon DocumentDB (serverlos).
- Wenn Sie in der Regel die Instance-Klasse Ihrer bereitgestellten Instances in Erwartung einer besonders hohen oder niedrigen Arbeitslast ändern, können Sie diese Erfahrung nutzen, um den entsprechenden serverlosen Kapazitätsbereich von DocumentDB abzuschätzen. Informationen zur Bestimmung der Speichergröße der bereitgestellten Amazon DocumentDB-Instances finden Sie unter. Kontingente für Instanzen

  Nehmen wir beispielsweise an, Sie verwenden die `db.r6g.4xlarge` Instance-Klasse, wenn Ihr Cluster eine hohe Arbeitslast hat. Diese Instanzklasse hat 128 GiB Speicher. Daher können Sie eine maximale DCU-Einstellung von 64 angeben, um eine serverlose Instanz einzurichten, die auf ungefähr dieselbe Kapazität skaliert werden kann. Das liegt daran, dass jede DCU ungefähr 2 GiB Arbeitsspeicher entspricht. Sie können einen etwas höheren Wert angeben, damit die Instance weiter skaliert werden kann, falls Ihre `db.r6g.4xlarge` Instance manchmal nicht über genügend Kapazität verfügt, um die Arbeitslast effektiv zu bewältigen.
- Wenn Sie eine Budgetobergrenze für Ihre Datenbanknutzung haben, wählen Sie einen Wert, der innerhalb dieser Obergrenze bleibt, auch wenn alle Ihre serverlosen Instances ständig mit maximaler Kapazität laufen. Denken Sie daran, dass, wenn Sie n serverlose Instances in Ihrem Cluster haben, die theoretische maximale serverlose Kapazität, die der Cluster jederzeit verbrauchen kann, das N-fache der maximalen DCU-Einstellung für den Cluster beträgt. (Der tatsächlich verbrauchte Betrag ist möglicherweise geringer, wenn beispielsweise einige Reader unabhängig vom Writer skalieren.)
- Wenn Sie serverlose Reader-Instances verwenden, um einen Teil der schreibgeschützten Arbeitslast von der Writer-Instance auszulagern, können Sie möglicherweise eine niedrigere Einstellung für die maximale Kapazität wählen. Sie tun dies, um zu berücksichtigen, dass nicht jede Reader-Instanz so hoch skaliert werden muss, als ob der Cluster nur eine einzige Instanz enthält.
- Angenommen, Sie möchten sich vor übermäßiger Auslastung aufgrund falsch konfigurierter Datenbankparameter oder ineffizienter Abfragen in Ihrer Anwendung schützen. In diesem Fall können Sie eine versehentliche Überbeanspruchung vermeiden, indem Sie eine maximale Kapazitätseinstellung wählen, die niedriger ist als die absolut höchste, die Sie festlegen können.
- Wenn Spitzen aufgrund realer Benutzeraktivitäten zwar selten sind, aber dennoch auftreten, können Sie diese Situationen bei der Auswahl der maximalen Kapazitätseinstellung berücksichtigen. Wenn die Priorität darin besteht, dass die Anwendung weiterhin mit voller Leistung

und Skalierbarkeit ausgeführt wird, können Sie eine maximale Kapazitätseinstellung angeben, die höher ist, als Sie bei normaler Auslastung beobachten. Wenn es akzeptabel ist, dass die Anwendung bei sehr extremen Aktivitätsspitzen mit reduziertem Durchsatz ausgeführt wird, können Sie eine etwas niedrigere maximale Kapazitätseinstellung wählen. Stellen Sie sicher, dass Sie eine Einstellung auswählen, die immer noch über genügend Speicher- und CPU-Ressourcen verfügt, damit die Anwendung weiterhin ausgeführt wird.

- Wenn Sie Einstellungen in Ihrem Cluster aktivieren, die den Speicherverbrauch für jede Instance erhöhen, berücksichtigen Sie diesen Speicher, wenn Sie den maximalen DCU-Wert festlegen. Zu diesen Einstellungen gehören die Einstellungen für Performance Insights und globale Cluster. Stellen Sie sicher, dass der maximale DCU-Wert es den serverlosen Instances ermöglicht, ausreichend zu skalieren, um die Arbeitslast zu bewältigen, wenn diese Funktionen verwendet werden. Informationen zur Behebung von Problemen, die durch die Kombination aus einer niedrigen maximalen DCU-Einstellung und Amazon DocumentDB-Funktionen verursacht werden, die einen Speicheraufwand verursachen, finden Sie unter Vermeiden von Fehlern aufgrund von zu wenig Speicher (unten).
- Verwenden Sie `MaxCapacity` für die angegebenen Funktionen mindestens das folgende Minimum (Änderungen vorbehalten):
  - Serverlose Instanzerstellung auf einem Cluster mit einem großen Datenvolumen — 2 DCUs (dazu gehört auch die serverlose Instanzerstellung im Rahmen einer Cluster-Wiederherstellung.)
- Bestimmte Instance-Limits werden von der aktuellen Kapazität der Instance bestimmt, z. B. Verbindungslimit, Cursorlimit und Limit für offene Transaktionen. Achten Sie bei der Auswahl des `MaxCapacity` Werts für Ihre Arbeitslast darauf, diese Instance-Limits zu berücksichtigen, um zu vermeiden, dass Sie durch eines dieser Limits zu Engpässen geraten. Weitere Informationen finden Sie unter Grenzwerte für serverlose Amazon DocumentDB-Instances.

Anweisungen zum Ändern der Skalierungskonfiguration eines Clusters finden Sie unter. Verwaltung von Amazon DocumentDB serverlos

Vermeiden von Fehlern aufgrund von zu wenig Speicher

Wenn eine Ihrer serverlosen DocumentDB-Instances ständig die Grenze ihrer maximalen Kapazität erreicht, weist Amazon DocumentDB auf diesen Zustand hin, indem es der Instance den Status „Incompatible-parameters“ zuweist. Während die Instance den Status „Incompatible-Parameters“ hat, sind einige Operationen blockiert. Beispielsweise können Sie die Engine-Version nicht aktualisieren. Weitere Informationen zum Status einer Amazon DocumentDB-Instance finden Sie unter. Überwachung des Status einer Amazon DocumentDB-Instance

In der Regel geht Ihre Instance in diesen Status über, wenn sie aufgrund von Fehlern aufgrund von Speichermangel häufig neu gestartet wird. Amazon DocumentDB zeichnet ein Ereignis auf, wenn diese Art von Neustart stattfindet. Informationen zum Anzeigen von Ressourcenereignissen finden Sie unterAmazon DocumentDB DocumentDB-Ereignisse anzeigen. Ein ungewöhnlich hoher Speicherverbrauch kann aufgrund des Overheads beim Aktivieren von Einstellungen wie Performance Insights auftreten. Dies kann auch auf eine hohe Arbeitslast Ihrer Instance oder auf die Verwaltung der Metadaten zurückzuführen sein, die mit einer großen Anzahl von Schemaobjekten verknüpft sind.

Wenn der Speicherdruck geringer wird, sodass die Instance ihre maximale Kapazität nicht sehr oft erreicht, ändert Amazon DocumentDB den Instance-Status automatisch wieder auf Verfügbar.

Zur Wiederherstellung nach diesem Zustand können Sie einige oder alle der folgenden Aktionen ausführen:

- Erhöhen Sie die untere Kapazitätsgrenze für serverlose Instances, indem Sie den Mindestwert für die DocumentDB-Kapazitätseinheit (DCU) für den Cluster ändern. Dadurch werden Probleme vermieden, bei denen eine inaktive Datenbank auf eine Kapazität mit weniger Speicher herunterskaliert wird, als für die in Ihrem Cluster aktivierten Funktionen benötigt wird. Nachdem Sie die DCU-Einstellungen für den Cluster geändert haben, starten Sie die serverlose Instance neu. Dadurch wird evaluiert, ob Amazon DocumentDB den Status wieder auf Verfügbar zurücksetzen kann.
- Erhöhen Sie die obere Kapazitätsgrenze für serverlose Instances, indem Sie den maximalen DCU-Wert für den Cluster ändern. Auf diese Weise werden Probleme vermieden, bei denen eine ausgelastete Datenbank nicht auf eine Kapazität mit genügend Speicher für die in Ihrem Cluster aktivierten Funktionen und die Datenbank-Workload hochskaliert werden kann. Nachdem Sie die DCU-Einstellungen für den Cluster geändert haben, starten Sie die serverlose Instance neu. Dadurch wird evaluiert, ob Amazon DocumentDB den Status wieder auf Verfügbar zurücksetzen kann.
- Deaktivieren Sie Konfigurationseinstellungen, die Speicher-Overhead erfordern. Nehmen wir zum Beispiel an, Sie haben eine Funktion wie Performance Insights aktiviert, verwenden sie aber nicht. Wenn ja, können Sie sie ausschalten. Oder Sie können die Mindest- und Höchstkapazitätswerte für den Cluster höher anpassen, um dem von diesen Featuretypen verwendeten Arbeitsspeicher Rechnung zu tragen. Richtlinien zur Auswahl der minimalen und maximalen Kapazitätseinstellungen finden Sie unter Auswahl des Skalierungskapazitätsbereichs für einen serverlosen DocumentDB-Cluster.

- Reduzieren Sie die Arbeitslast der Instanz. Sie können dem Cluster beispielsweise Reader-Instanzen hinzufügen, um die Last von schreibgeschützten Abfragen auf mehrere Instanzen zu verteilen.

Warum wird meine serverlose Instance nicht herunterskaliert?

In einigen Fällen wird DocumentDB serverless nicht auf die Mindestkapazität herunterskaliert, auch wenn die Datenbank nicht belastet wird. Dies kann aus einem der folgenden Gründe geschehen:

- Performance Insights kann die Ressourcennutzung erhöhen und verhindern, dass die Datenbank auf die Mindestkapazität herunterskaliert wird. Nachstehend sind einige dieser Features aufgeführt:
- Wenn eine Reader-Instanz nicht auf das Minimum herunterskaliert wird und die gleiche oder eine höhere Kapazität als die Writer-Instanz hat, überprüfen Sie die Prioritätsstufe der Reader-Instance. Serverlose DocumentDB-Leser-Instanzen auf Stufe 0 oder 1 haben eine Mindestkapazität, die mindestens so hoch ist wie die Writer-Instanz. Ändern Sie die Prioritätsstufe der Reader-Instance in 2 oder höher, sodass sie unabhängig von der Writer-Instance hoch- und herunterskaliert wird. Weitere Informationen finden Sie unter Serverlose Skalierung von Amazon DocumentDB.
- Hohe Datenbank-Workloads können die Ressourcennutzung erhöhen.
- Große Datenbank-Volumes können die Ressourcennutzung erhöhen. Amazon DocumentDB verwendet Speicher- und CPU-Ressourcen für die Clusterverwaltung. Amazon DocumentDB benötigt mehr CPU und Arbeitsspeicher, um Cluster mit größeren Datenbankvolumina zu verwalten. Wenn die Mindestkapazität Ihres Clusters unter der für die Clusterverwaltung erforderlichen Mindestkapazität liegt, skaliert Ihr Cluster nicht auf die Mindestkapazität herunter.
- Wartungsaktivitäten im Hintergrund können den Ressourcenverbrauch in regelmäßigen Abständen erhöhen.

Wenn die Datenbank immer noch nicht auf die konfigurierte Mindestkapazität herunterskaliert werden kann, beenden Sie die Datenbank und starten Sie sie neu, um alle Speicherfragmente zurückzugewinnen, die sich im Laufe der Zeit möglicherweise angesammelt haben.

> **Note**
>
> Das Stoppen und Starten einer Datenbank führt zu Ausfallzeiten. Tun Sie dies nur, wenn es nötig ist.

Überwachung von Amazon DocumentDB (serverlos)

Weitere Informationen zur Überwachung in Amazon DocumentDB finden Sie unter. Überwachung und Protokollierung in Amazon DocumentDB

Topics

- Nicht genügend Arbeitsspeicher: Status inkompatibler Parameter
- Amazon-Metriken für serverlose DocumentDB CloudWatch
- Überwachung der serverlosen DocumentDB-Leistung mit Performance Insights

Nicht genügend Arbeitsspeicher: Status inkompatibler Parameter

Wenn eine Ihrer serverlosen Instances ständig das Limit ihrer maximalen Kapazität erreicht, weist Amazon DocumentDB auf diesen Zustand hin, indem es der Instance den Status „Incompatible-parameters“ zuweist. Weitere Informationen finden Sie unter Vermeiden von Fehlern aufgrund von zu wenig Speicher.

Amazon-Metriken für serverlose DocumentDB CloudWatch

Weitere Informationen zur Verwendung CloudWatch mit Amazon DocumentDB finden Sie unter. Überwachung von Amazon DocumentDB mit CloudWatch

Sie können Ihre serverlosen Instances in einsehen CloudWatch , um die von jeder Instance verbrauchte Kapazität anhand der `ServerlessDatabaseCapacity` Metrik zu überwachen. Sie können auch alle standardmäßigen CloudWatch DocumentDB-Metriken wie `DatabaseConnections` und -Abfragen überwachen. Die vollständige Liste der CloudWatch Metriken, die Sie für Amazon DocumentDB überwachen können, finden Sie unter. Überwachung von Amazon DocumentDB mit CloudWatch Es ist wichtig, die folgenden Metriken CloudWatch auf Instance-Ebene zu überwachen, damit Sie verstehen, wie Ihre serverlosen DocumentDB-Instances nach oben und unten skaliert werden. Alle diese Metriken werden jede Sekunde berechnet. Auf diese Weise können Sie den aktuellen Status Ihrer serverlosen Instances überwachen. Sie können Alarme einrichten, um Sie zu benachrichtigen, wenn sich eine serverlose Instance einem Schwellenwert für kapazitätsbezogene Metriken nähert. Sie können feststellen, ob die minimalen und maximalen Kapazitätseinstellungen angemessen sind oder ob Sie sie anpassen müssen. Sie können bestimmen, worauf Sie sich konzentrieren müssen, um die Effizienz Ihrer Datenbank zu optimieren.

- **`ServerlessDatabaseCapacity`**— Als Metrik auf Instance-Ebene gibt sie die Anzahl der DCUs an, die durch die aktuelle Instance-Kapazität repräsentiert werden. Als Metrik auf Clusterebene stellt sie den Durchschnitt der `ServerlessDatabaseCapacity` Werte aller serverlosen DocumentDB-Instances im Cluster dar.
- **`DCUUtilization.`**— Diese Metrik ist neu in DocumentDB serverless. Dieser Wert wird als Prozentsatz dargestellt. Sie wird berechnet als der Wert der `ServerlessDatabaseCapacity` Metrik geteilt durch den maximalen DCU-Wert des Clusters. Beachten Sie die folgenden Richtlinien, um diese Metrik zu interpretieren und Maßnahmen zu ergreifen:
  - Wenn sich diese Metrik einem Wert von nähert`100.0`, wurde die Instanz so hoch wie möglich skaliert. Erwägen Sie, die maximale DCU-Einstellung für den Cluster zu erhöhen. Auf diese Weise können sowohl Writer- als auch Reader-Instances auf eine höhere Kapazität skaliert werden.
  - Nehmen wir an, dass eine Reader-Instanz aufgrund einer Workload, die nur zum Lesen verwendet wird, sich einem Wert `DCUUtilization` von nähert`100.0`, während die Writer-Instanz nicht annähernd ihre maximale Kapazität erreicht hat. In diesem Fall sollten Sie erwägen, dem Cluster weitere Reader-Instances hinzuzufügen. Auf diese Weise können Sie den schreibgeschützten Teil der Arbeitslast auf mehrere Instanzen verteilen und so die Belastung jeder Reader-Instanz reduzieren.
  - Angenommen, Sie führen eine Produktionsanwendung aus, bei der Leistung und Skalierbarkeit die Hauptüberlegungen sind. In diesem Fall können Sie den maximalen DCU-Wert für den Cluster auf einen hohen Wert festlegen. Ihr Ziel ist es, dass die `DCUUtilization` Metrik immer unter 100,0 liegt. Bei einem hohen maximalen DCU-Wert können Sie sicher sein, dass genügend Platz vorhanden ist, falls es zu unerwarteten Spitzen bei der Datenbankaktivität kommen sollte. Berechnet wird Ihnen nur die tatsächlich verbrauchte Datenbankkapazität.
- **`CPUUtilization`**— Diese Metrik wird in DocumentDB serverless anders interpretiert als in bereitgestellten Instanzen. Für serverlose DocumentDB ist dieser Wert ein Prozentsatz, der als die Menge der aktuell verwendeten CPU berechnet wird, geteilt durch die CPU-Kapazität, die unter dem maximalen DCU-Wert des Clusters verfügbar ist. Amazon DocumentDB überwacht diesen Wert automatisch und skaliert Ihre serverlose Instance, wenn die Instance konstant einen hohen Anteil ihrer CPU-Kapazität nutzt.

  Wenn sich diese Metrik einem Wert von nähert`100.0`, hat die Instance ihre maximale CPU-Kapazität erreicht. Erwägen Sie, die maximale DCU-Einstellung für den Cluster zu erhöhen. Wenn sich diese Metrik `100.0` auf einer Reader-Instance einem Wert von nähert, sollten Sie erwägen, dem Cluster weitere Reader-Instances hinzuzufügen. Auf diese Weise können Sie den

schreibgeschützten Teil der Arbeitslast auf mehrere Instanzen verteilen und so die Belastung jeder Reader-Instance reduzieren.

- **`FreeableMemory`**— Dieser Wert stellt die Menge an unbenutztem Speicher dar, die verfügbar ist, wenn die serverlose DocumentDB-Instanz auf ihre maximale Kapazität skaliert wird. Für jede DCU, deren aktuelle Kapazität unter der maximalen Kapazität liegt, erhöht sich dieser Wert um etwa 2 GiB. Daher nähert sich diese Metrik erst dann Null, wenn die Instance so hoch wie möglich skaliert wird.

  Nähert sich diese Metrik einem Wert von Null, wurde die Instanz so weit wie möglich hochskaliert und nähert sich der Grenze ihres verfügbaren Speichers. Erwägen Sie, die maximale DCU-Einstellung für den Cluster zu erhöhen. Wenn sich diese Metrik auf einer Reader-Instance einem Wert von Null nähert, sollten Sie erwägen, dem Cluster weitere Reader-Instances hinzuzufügen. Auf diese Weise kann der schreibgeschützte Teil der Arbeitslast auf mehrere Instanzen verteilt werden, wodurch der Speicherverbrauch auf jeder Reader-Instance reduziert wird.

- **`TempStorageIops`**— Die Anzahl der IOPS, die auf dem lokalen Speicher vorgenommen wurden, der an die Instanz angehängt ist. Dabei sind IOPS für Lese- und Schreibvorgänge enthalten. Diese Metrik stellt eine Zählung dar und wird einmal pro Sekunde gemessen. Dies ist eine neue Metrik für DocumentDB serverless. Details hierzu finden Sie unter Überwachung von Amazon DocumentDB mit CloudWatch.

- **`TempStorageThroughput`**— Die Datenmenge, die zum und vom lokalen Speicher übertragen wird, der der Instanz zugeordnet ist. Diese Metrik wird in Byte angegeben und einmal pro Sekunde gemessen. Dies ist eine neue Metrik für DocumentDB serverless. Details hierzu finden Sie unter Überwachung von Amazon DocumentDB mit CloudWatch.

In der Regel wird die Skalierung für serverlose DocumentDB-Instances hauptsächlich durch Speicherauslastung und CPU-Aktivität verursacht. Die `TempStorageIops` und `TempStorageThroughput` -Metriken können Ihnen helfen, die seltenen Fälle zu diagnostizieren, in denen Netzwerkaktivitäten für Übertragungen zwischen Ihrer Instance und lokalen Speichergeräten für unerwartete Kapazitätserhöhungen verantwortlich sind. Wenn Sie andere Netzwerkaktivitäten überwachen möchten, können Sie diese vorhandenen Metriken verwenden:

- `NetworkReceiveThroughput`
- `NetworkThroughput`
- `NetworkTransmitThroughput`
- `StorageNetworkReceiveThroughput`

- `StorageNetworkThroughput`
- `StorageNetworkTransmitThroughput`

Wie gelten serverlose CloudWatch DocumentDB-Metriken für Ihre AWS Rechnung

Die serverlosen DocumentDB-Gebühren auf Ihrer AWS Rechnung werden auf der Grundlage derselben `ServerlessDatabaseCapacity` Kennzahl berechnet, die Sie überwachen können. Der Abrechnungsmechanismus kann vom berechneten CloudWatch Durchschnitt für diese Metrik abweichen, wenn Sie die serverlose DocumentDB-Kapazität nur für einen Teil einer Stunde nutzen. Er kann auch abweichen, wenn die CloudWatch Metrik aufgrund von Systemproblemen kurzzeitig nicht verfügbar ist. Daher wird DCU-hours auf Ihrer Rechnung möglicherweise ein etwas anderer Wert von angezeigt, als wenn Sie die Zahl selbst anhand des `ServerlessDatabaseCapacity` Durchschnittswerts berechnen würden.

Beispiele für Amazon CloudWatch CLI-Befehle für serverlose DocumentDB-Metriken

Die folgenden AWS CLI Beispiele zeigen, wie Sie die wichtigsten CloudWatch Metriken im Zusammenhang mit DocumentDB serverless überwachen können. Ersetzen Sie in jedem Fall die `Value=` Zeichenfolge für den `--dimensions` Parameter durch den Bezeichner Ihrer eigenen serverlosen DocumentDB-Instanz.

Im folgenden Linux-Beispiel werden die Mindest-, Höchst- und Durchschnittskapazitätswerte für eine Instance angezeigt, gemessen alle 10 Minuten über einen Zeitraum von einer Stunde. Die Linux-Datumsbefehle geben die Start- und Endzeiten relativ zum aktuellen Datum und zur aktuellen Uhrzeit an. Die `sort_by`-Funktion im `--query`-Parameter sortiert die Ergebnisse chronologisch basierend auf dem Feld `Timestamp`.

```
aws cloudwatch get-metric-statistics \
    --metric-name "ServerlessDatabaseCapacity" \
    --start-time "$(date -d '1 hour ago')" \
    --end-time "$(date -d 'now')" \
    --period 600 \
    --namespace "AWS/DocDB" \
    --statistics Minimum Maximum Average \
    --dimensions Name=DBInstanceIdentifier,Value=my_instance \
    --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' \
    --output table
```

Das folgende Linux-Beispiel zeigt die Überwachung der Kapazität einer Instance in einem Cluster. Es misst die minimale, maximale und durchschnittliche Kapazitätsauslastung einer Instance. Die Messungen werden einmal pro Stunde über einen Zeitraum von drei Stunden durchgeführt. In diesen Beispielen wird die `DCUUtilization` Metrik verwendet, die einen Prozentsatz der Obergrenze für DCUs `ServerlessDatabaseCapacity` darstellt, anstatt eine feste Anzahl von DCUs darzustellen. Auf diese Weise müssen Sie die tatsächlichen Zahlen für die minimalen und maximalen DCU-Werte im Kapazitätsbereich nicht kennen. Sie können Prozentsätze zwischen 0 und 100 sehen.

```
aws cloudwatch get-metric-statistics \
    --metric-name "DCUUtilization" \
    --start-time "$(date -d '3 hours ago')" \
    --end-time "$(date -d 'now')" \
    --period 3600 \
    --namespace "AWS/DocDB" \
    --statistics Minimum Maximum Average \
    --dimensions Name=DBInstanceIdentifier,Value=my_instance \
    --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' \
    --output table
```

Im folgenden Linux-Beispiel werden ähnliche Messungen wie die vorherigen ausgeführt. In diesem Fall gelten die Messungen für die `CPUUtilization`-Metrik Die Messungen werden alle zehn Minuten über einen Zeitraum von einer Stunde durchgeführt. Die Zahlen stellen den Prozentsatz der verwendeten verfügbaren CPU dar, basierend auf den CPU-Ressourcen, die bis zur maximalen Kapazitätseinstellung für die Instance verfügbar sind.

```
aws cloudwatch get-metric-statistics \
    --metric-name "CPUUtilization" \
    --start-time "$(date -d '1 hour ago')" \
    --end-time "$(date -d 'now')" \
    --period 600 \
    --namespace "AWS/DocDB" \
    --statistics Minimum Maximum Average \
    --dimensions Name=DBInstanceIdentifier,Value=my_instance \
    --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' \
    --output table
```

Im folgenden Linux-Beispiel werden ähnliche Messungen wie die vorherigen ausgeführt. In diesem Fall gelten die Messungen für die `FreeableMemory`-Metrik Die Messungen werden alle zehn Minuten über einen Zeitraum von einer Stunde durchgeführt.

```
aws cloudwatch get-metric-statistics \
    --metric-name "FreeableMemory" \
    --start-time "$(date -d '1 hour ago')" \
    --end-time "$(date -d 'now')" \
    --period 600 \
    --namespace "AWS/DocDB" \
    --statistics Minimum Maximum Average \
    --dimensions Name=DBInstanceIdentifier,Value=my_instance \
    --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' \
    --output table
```

Überwachung der serverlosen DocumentDB-Leistung mit Performance Insights

Sie können Performance Insights verwenden, um die Leistung von serverlosen DocumentDB-Instances zu überwachen. Informationen zu Performance-Insights-Verfahren finden Sie unter Überwachung mit Performance Insights.

Die folgenden neuen Performance Insights-Leistungsindikatoren gelten für serverlose DocumentDB-Instances:

- **`os.general.serverlessDBCapacity`**— Die aktuelle Kapazität der Instance in DCUs. Der Wert entspricht der `ServerlessDatabaseCapacity` CloudWatch Metrik für die Instanz.
- **`os.general.dcuUtilization`**— Der Prozentsatz der aktuellen Kapazität gegenüber der maximal konfigurierten Kapazität. Der Wert entspricht der `DCUUtilization` CloudWatch Metrik für die Instance.
- **`os.general.maxConfiguredDcu`**— Die maximale Kapazität, die Sie für diese serverlose DocumentDB-Instance konfiguriert haben. Sie wird in DCUs gemessen.
- **`os.general.minConfiguredDcu`**— Die Mindestkapazität, die Sie für diese serverlose DocumentDB-Instance konfiguriert haben. Sie wird in DCUs gemessen.

Eine vollständige Liste der Performance-Insights-Zähler finden Sie unter Leistungseinblicke für Zählerkennzahlen.

Wenn vCPU-Werte für eine serverlose DocumentDB-Instance in Performance Insights angezeigt werden, stellen diese Werte Schätzungen dar, die auf dem DCU-Wert für die Instance basieren. Im Standardintervall von einer Minute werden alle fraktionierten vCPU-Werte auf die nächste ganze Zahl aufgerundet. Für längere Intervalle ist der angezeigte vCPU-Wert der Durchschnitt der ganzzahligen vCPU-Werte für jede Minute.

Verwenden von Amazon DocumentDB Elastic Clusters

Die elastischen Amazon DocumentDB-Cluster unterstützen Workloads mit Millionen von reads/writes pro Sekunde und Petabyte an Speicherkapazität. Elastische Cluster vereinfachen auch die Interaktion von Entwicklern mit Amazon DocumentDB, da Instances nicht mehr ausgewählt, verwaltet oder aktualisiert werden müssen.

> **Note**
>
> Elastische Cluster verwenden ein MongoDB 5.0-kompatibles Wire-Protokoll. Sie werden in der Amazon DocumentDB-Engine Version 8.0 nicht unterstützt und sind in den Regionen China oder nicht verfügbar. AWS GovCloud

Elastische Amazon DocumentDB-Cluster wurden erstellt, um:

- Bieten Sie eine Lösung für Kunden, die nach einer Datenbank suchen, die eine praktisch unbegrenzte Skalierbarkeit mit umfangreichen Abfragefunktionen und MongoDB-API-Kompatibilität bietet.
- Bieten Sie Ihren Kunden höhere Verbindungslimits und reduzieren Sie Ausfallzeiten durch Patches.
- Investieren Sie weiterhin in eine Cloud-native, elastische und branchenführende Architektur für JSON-Workloads.

Topics

- Anwendungsfälle für elastische Cluster
- Vorteile elastischer Cluster
- Verfügbarkeit von Regionen und Versionen des Elastic Clusters
- Einschränkungen
- Elastische Amazon DocumentDB-Cluster: So funktioniert es
- Erste Schritte mit Amazon DocumentDB Elastic Clusters
- Bewährte Methoden für Amazon DocumentDB Elastic Cluster
- Elastische Amazon DocumentDB-Cluster verwalten

- Datenverschlüsselung im Ruhezustand für elastische Amazon DocumentDB-Cluster
- Service-linked Rollen in elastischen Clustern

Anwendungsfälle für elastische Cluster

Dokumentdatenbanken sind nützlich für Workloads, die ein flexibles Schema für eine schnelle, iterative Entwicklung benötigen. Beispiele für Amazon DocumentDB-Anwendungsfälle finden Sie unterAnwendungsfälle für Dokumentendatenbanken.

Im Folgenden finden Sie einige Beispiele für Anwendungsfälle, für die elastische Cluster erhebliche Vorteile bieten können:

Benutzerprofile

Da Dokumentdatenbanken über ein flexibles Schema verfügen, können sie Dokumente mit unterschiedlichen Attributen und Datenwerten maßstabsgetreu speichern. Elastische Cluster sind eine praktische Lösung für Online-Profile, in denen verschiedene Benutzer unterschiedliche Arten von Informationen bereitstellen. Angenommen, Ihre Anwendungen unterstützen Hunderte Millionen von Benutzerprofilen. Sie können elastische Cluster verwenden, um solche Anwendungen zu unterstützen, da sie hoch- und herunterskaliert werden können, um Millionen von Schreib- und Lesevorgängen in diesen Benutzerprofilen zu unterstützen. Sie können das System auch außerhalb der Spitzenzeiten herunterskalieren, um die Kosten zu senken.

Inhaltsmanagement und historische Aufzeichnungen

Um Inhalte effektiv zu verwalten, müssen Sie in der Lage sein, Inhalte aus einer Vielzahl von Quellen zu sammeln, zu aggregieren und dann an den Client zu liefern. Aufgrund ihres flexiblen Schemas sind Dokumentdatenbanken ideal für die Erfassung und Speicherung jeglicher Art von Daten. Sie können sie verwenden, um neue Arten von Inhalten zu erstellen und zu integrieren, einschließlich benutzergenerierter Inhalte wie Bilder, Kommentare und Videos. Im Laufe der Zeit benötigt Ihre Datenbank möglicherweise mehr Speicherplatz. Mit elastischen Clustern können Sie Ihre Daten auf mehr Speichervolumes verteilen, sodass Sie Petabyte an Daten in einem einzigen Cluster speichern können.

Vorteile elastischer Cluster

AWS Serviceintegration

Elastische Amazon DocumentDB-Cluster lassen sich auf die gleiche Weise in andere AWS Dienste integrieren, wie es Amazon DocumentDB tut:

- Migration — Sie können AWS Database Migration Service (DMS) verwenden, um von MongoDB und anderen relationalen Datenbanken zu elastischen Amazon DocumentDB-Clustern zu migrieren.
- Überwachung — Sie können den Zustand und die Leistung Ihres Elastic Clusters mithilfe von Amazon überwachen. CloudWatch
- Sicherheit — Sie können die Authentifizierung und Autorisierung über AWS Identity and Access Management (IAM) einrichten, um Ihre Elastic Cluster-Ressourcen zu verwalten, z. B. einen Elastic Cluster zu erstellen und zu aktualisieren, und Amazon VPC für sichere VPC-only Verbindungen verwenden. Die Anmeldung mit IAM-Authentifizierung für elastische Cluster wird nicht unterstützt.
- Datenmanagement — Sie können from/to andere AWS Dienste wie Amazon S3, Amazon Redshift und Amazon Service verwenden, AWS Glue um Daten zu importieren und zu exportieren. OpenSearch

Verfügbarkeit von Regionen und Versionen des Elastic Clusters

Regionale Verfügbarkeit für elastische Cluster

Die folgende Tabelle zeigt, AWS-Regionen wo Amazon DocumentDB Elastic Clusters derzeit verfügbar sind, und den Endpunkt für jede Region.

Name der Region	Region	Availability Zones
USA Ost (Nord-Virginia)	`us-east-1`	5
USA Ost (Ohio)	`us-east-2`	3
USA West (Oregon)	`us-west-2`	3
Asien-Pazifik (Hongkong)	`ap-east-1`	3
Asien-Pazifik (Mumbai)	`ap-south-1`	3

Name der Region	Region	Availability Zones
Asien-Pazifik (Seoul)	ap-northeast-2	3
Asien-Pazifik (Singapur)	ap-southeast-1	3
Asien-Pazifik (Sydney)	ap-southeast-2	3
Asien-Pazifik (Tokio)	ap-northeast-1	3
Kanada (Zentral)	ca-central-1	3
Südamerika (São Paulo)	sa-east-1	3
Europa (Frankfurt)	eu-central-1	3
Europa (Irland)	eu-west-1	3
Europa (London)	eu-west-2	3
Europa (Milan)	eu-south-1	3
Europa (Paris)	eu-west-3	3

Verfügbarkeit der Versionen

Elastische Cluster unterstützen das MongoDB 5.0-kompatible Wire-Protokoll. Informationen zu Unterschieden zwischen instanzbasierten Amazon DocumentDB 4.0-Clustern und elastischen Clustern finden Sie unter. Funktionale Unterschiede zwischen Amazon DocumentDB 4.0 und Elastic Clusters Hinweis: Elastic Clusters werden in Amazon DocumentDB 8.0 nicht unterstützt

Einschränkungen

Elastische Clusterverwaltung

Die folgenden Funktionen und Funktionen der Clusterverwaltung werden in dieser Version nicht unterstützt:

- Benutzerdefinierter Port (nur 27017 wird unterstützt)

- Möglichkeit, IAM-Benutzer und -Rollen zur Authentifizierung in der Datenbank zu verwenden
- Fähigkeit, globale Cluster zu erstellen
- Bestehende Amazon DocumentDB-Ereignisse und Abonnieren von Ereignissen
- Reichweiten-Sharding
- Bestehende Sammlung teilen
- Multi-field Schlüssel teilen
- Shard-Schlüssel ändern
- Point-in-time wiederherstellen
- Klonen
- Performance Insights

> **Note**
>
> Hinweise zu Quoten für elastische Cluster finden Sie unterAmazon DocumentDB-Kontingente.

Abfrage- und Schreibvorgänge

Die folgenden Befehle und Funktionen für Abfrage- und Schreibvorgänge werden in dieser Version nicht unterstützt:

- DDL-Befehle bei Skalierungsvorgängen
- Profiler
- Parametergruppen
- AWS Config
- AWS Backup

Sammlungs- und Indexverwaltung

Die folgenden Funktionen zur Sammlungs- und Indexverwaltung werden in dieser Version nicht unterstützt:

- Eindeutige Indizes
- Nicht vollständige Indizes
- Textindizes
- Vektor-Indizes
- Komprimierung von Dokumenten

Verwaltung und Diagnose

Die folgenden Verwaltungs- und Diagnosebefehle und -funktionen werden in dieser Version nicht unterstützt:

- AWS Secrets Manager
- Role-based-access-control (RBAC) benutzerdefinierte Rollen.
- Beim Herstellen einer Verbindung wird das Schreiben mit dem Wert 0 nicht unterstützt.
- Ändern von Subnetzen, die zu einer VPC gehören, die derzeit keinem vorhandenen Elastic Cluster zugewiesen ist.

Opt-in features

Die folgenden Amazon DocumentDB-Opt-In-Funktionen werden in dieser Version nicht unterstützt:

- ACID-Transaktionen
- DDL/DML Prüfung
- Change streams
- Befehle für Sitzungen

Elastische Amazon DocumentDB-Cluster: So funktioniert es

Die Themen in diesem Abschnitt enthalten Informationen zu den Mechanismen und Funktionen, die den elastischen Clustern von Amazon DocumentDB zugrunde liegen.

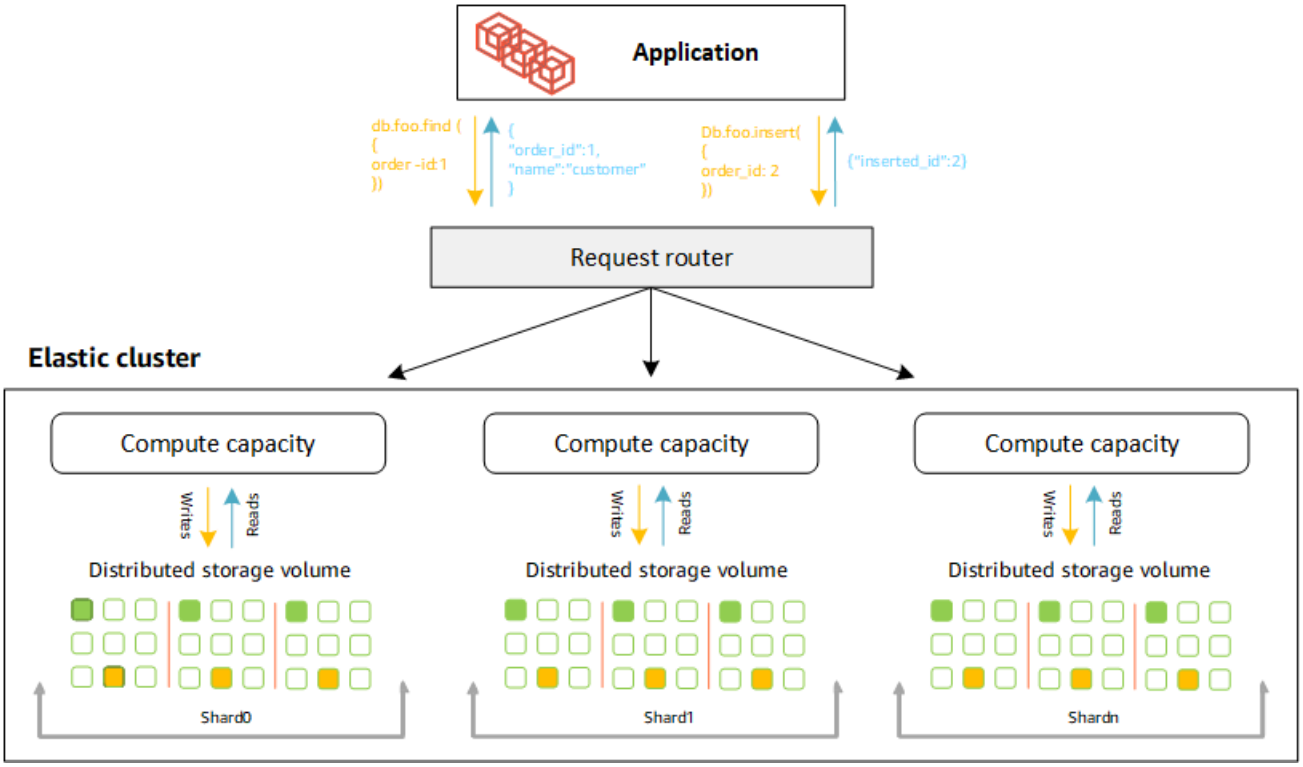
Topics

- Elastisches Cluster-Sharding von Amazon DocumentDB

- Elastische Cluster-Migration
- Elastische Cluster-Skalierung
- Elastische Cluster-Zuverlässigkeit
- Elastischer Cluster-Speicher und Verfügbarkeit
- Funktionale Unterschiede zwischen Amazon DocumentDB 4.0 und Elastic Clusters

Elastisches Cluster-Sharding von Amazon DocumentDB

Elastische Amazon DocumentDB-Cluster verwenden Hash-basiertes Sharding, um Daten in einem verteilten Speichersystem zu partitionieren. Sharding, auch Partitionierung genannt, teilt große Datensätze über mehrere Knoten hinweg in kleine Datensätze auf, sodass Sie Ihre Datenbank über vertikale Skalierungsgrenzen hinaus skalieren können. Elastische Cluster nutzen die Trennung oder „Entkopplung“ von Rechenleistung und Speicher in Amazon DocumentDB, sodass Sie unabhängig voneinander skalieren können. Anstatt Sammlungen neu zu partitionieren, indem kleine Datenblöcke zwischen Rechenknoten verschoben werden, kopieren elastische Cluster Daten effizient innerhalb des verteilten Speichersystems.



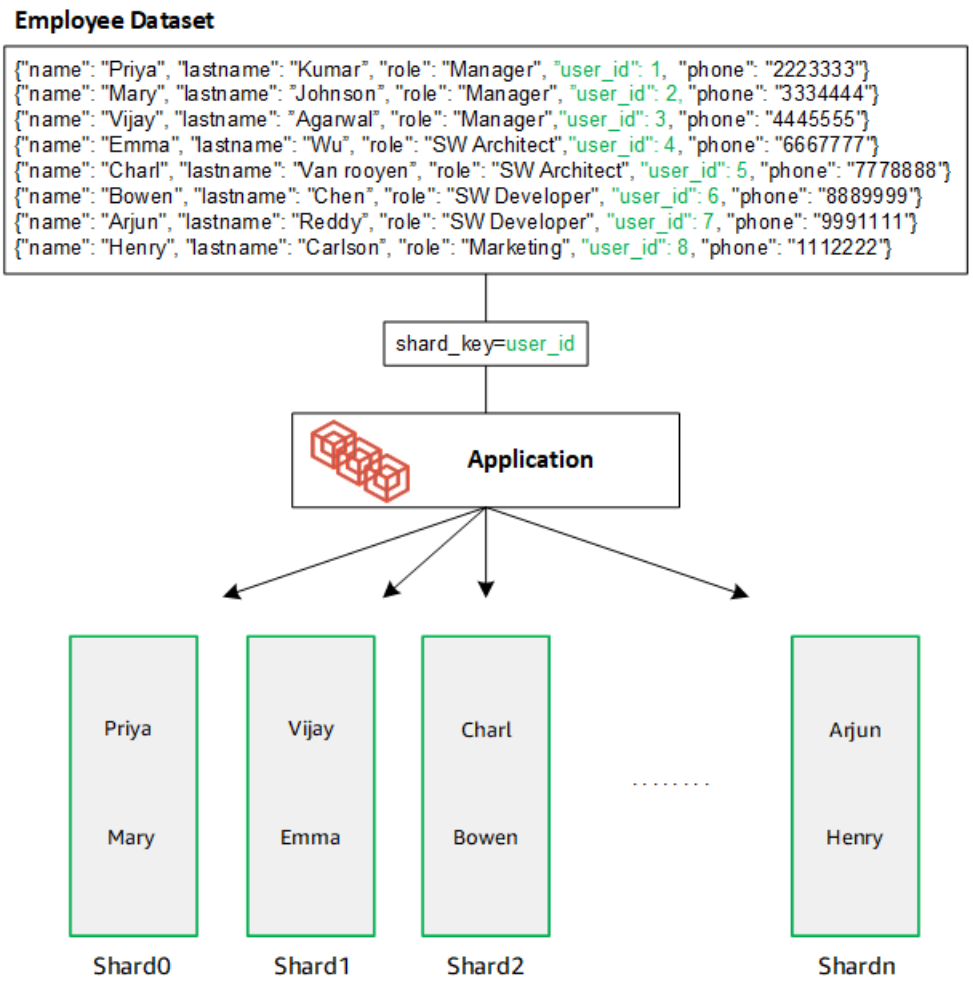
Shard-Definitionen

Definitionen der Shard-Nomenklatur:

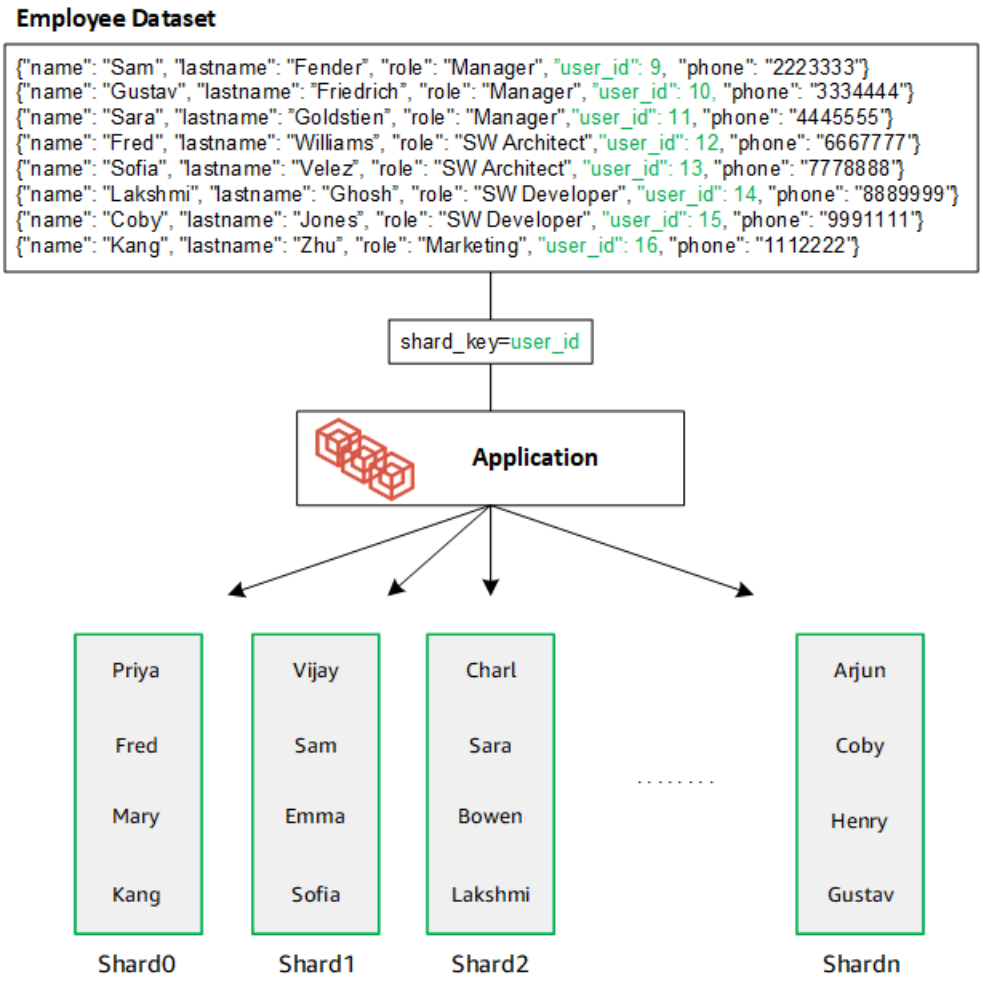
- Shard — Ein Shard bietet Rechenleistung für einen elastischen Cluster. Es wird eine einzige Writer-Instanz und 0—15 Read-Repliken haben. Standardmäßig hat ein Shard zwei Instanzen: eine Writer- und eine Single-Read-Replica. Sie können maximal 32 Shards konfigurieren und jede Shard-Instance kann maximal 64 vCPUs haben.
- Shard-Schlüssel — Ein Shard-Schlüssel ist ein Pflichtfeld in Ihren JSON-Dokumenten in geteilten Sammlungen, die elastische Cluster verwenden, um den Lese- und Schreibverkehr an den entsprechenden Shard zu verteilen.
- Sharded Collection — Eine Sharded Collection ist eine Sammlung, deren Daten in Datenpartitionen über einen elastischen Cluster verteilt sind.
- Partition — Eine Partition ist ein logischer Teil der geteilten Daten. Wenn Sie eine gemeinsame Sammlung erstellen, werden die Daten innerhalb jedes Shards automatisch auf der Grundlage des Shard-Schlüssels in Partitionen organisiert. Jeder Shard hat mehrere Partitionen.

Verteilen von Daten auf konfigurierte Shards

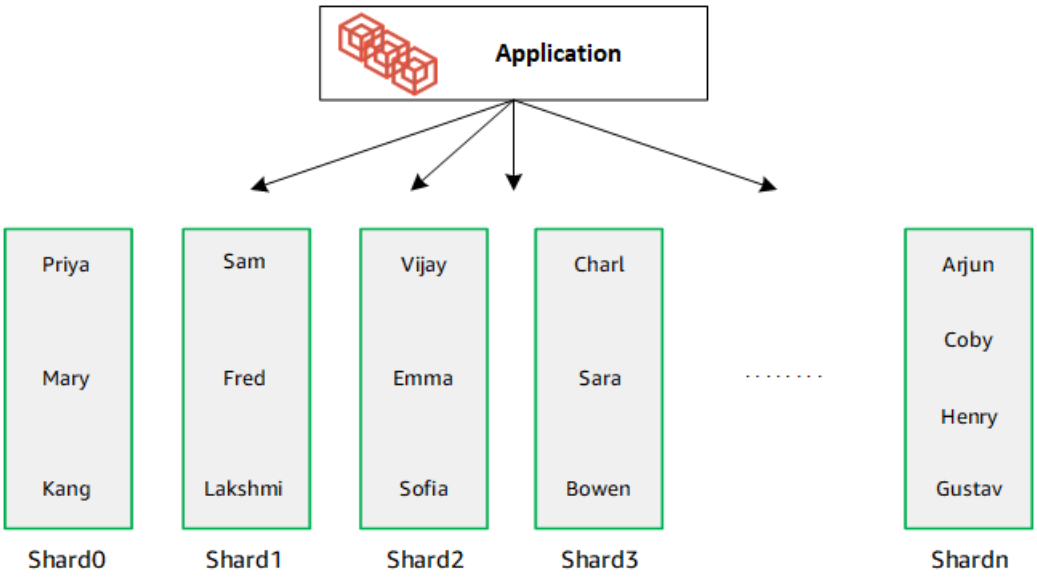
Erstellen Sie einen Shard-Schlüssel mit vielen eindeutigen Werten. Ein guter Shard-Schlüssel verteilt Ihre Daten gleichmäßig auf die zugrunde liegenden Shards, sodass Ihre Arbeitslast den besten Durchsatz und die beste Leistung erzielt. Das folgende Beispiel sind Mitarbeiternamendaten, die einen Shard-Schlüssel namens „user_id“ verwenden:



DocumentDB verwendet Hash-Sharding, um Ihre Daten auf die zugrunde liegenden Shards zu partitionieren. Zusätzliche Daten werden auf die gleiche Weise eingefügt und verteilt:



Wenn Sie Ihre Datenbank skalieren, indem Sie zusätzliche Shards hinzufügen, verteilt Amazon DocumentDB die Daten automatisch neu:



Elastische Cluster-Migration

Amazon DocumentDB unterstützt die Migration von MongoDB-Shard-Daten zu elastischen Clustern. Offline-, Online- und Hybrid-Migrationsmethoden werden unterstützt. Weitere Informationen finden Sie unter Migration und Aktualisierung von Amazon DocumentDB.

Elastische Cluster-Skalierung

Elastische Amazon DocumentDB-Cluster bieten die Möglichkeit, die Anzahl der Shards (Scale-Out) in Ihrem elastischen Cluster und die Anzahl der auf jeden Shard angewendeten vCPUs (Scale-Up) zu erhöhen. Sie können auch die Anzahl der Shards und die Rechenkapazität (vCPUs) nach Bedarf reduzieren.

Bewährte Methoden zur Skalierung finden Sie unter. Skalierung elastischer Cluster

> **Note**
>
> Cluster-level Skalierung ist ebenfalls verfügbar. Weitere Informationen finden Sie unter Skalieren von Amazon DocumentDB-Clustern.

Elastische Cluster-Zuverlässigkeit

Amazon DocumentDB wurde so konzipiert, dass es zuverlässig, robust und fehlertolerant ist. Um die Verfügbarkeit zu verbessern, setzen elastische Cluster zwei Knoten pro Shard ein, die in verschiedenen Availability Zones platziert sind. Amazon DocumentDB enthält mehrere automatische Funktionen, die es zu einer zuverlässigen Datenbanklösung machen. Weitere Informationen finden Sie unter Zuverlässigkeit von Amazon DocumentDB.

Elastischer Cluster-Speicher und Verfügbarkeit

Amazon DocumentDB-Daten werden auf einem Cluster-Volume gespeichert. Dabei handelt es sich um ein einzelnes, virtuelles Volume, das Solid-State-Laufwerke (SSDs) verwendet. Ein Cluster-Volume besteht aus sechs Kopien Ihrer Daten, die automatisch über mehrere Availability Zones in einer einzigen Region repliziert werden. AWS Diese Replikation trägt dazu bei, dass Ihre Daten sehr langlebig sind und weniger Datenverlust möglich ist. Sie trägt außerdem dazu bei, dass Ihr Cluster während eines Failovers besser verfügbar ist, da Kopien Ihrer Daten bereits in anderen Availability Zones vorhanden sind. Weitere Informationen zu Speicher, Hochverfügbarkeit und Replikation finden Sie unterAmazon DocumentDB: wie funktioniert es.

Funktionale Unterschiede zwischen Amazon DocumentDB 4.0 und Elastic Clusters

Die folgenden funktionalen Unterschiede bestehen zwischen Amazon DocumentDB 4.0 und Elastic Clusters.

- Ergebnisse von `top` und `collStats` sind durch Shards partitioniert. Bei gemeinsam genutzten Sammlungen werden die Daten auf mehrere Partitionen verteilt, und die `collStats` Berichte werden anhand der Partitionen aggregiert. `collScans`
- Die Sammlungsstatistiken von `top` und `collStats` für gemeinsam genutzte Sammlungen werden zurückgesetzt, wenn die Anzahl der Cluster-Shards geändert wird.
- Die integrierte Backup-Rolle unterstützt jetzt. `serverStatus` Aktion — Entwickler und Anwendungen mit Backup-Rolle können Statistiken über den Status des Amazon DocumentDB-Clusters sammeln.
- Das `SecondaryDelaySecs` Feld wird `slaveDelay` in der `replSetGetConfig` Ausgabe ersetzt.
- Der `hello` Befehl ersetzt `isMaster` — `hello` gibt ein Dokument zurück, das die Rolle des elastischen Clusters beschreibt.
- Der `$elemMatch` Operator in elastischen Clustern entspricht nur Dokumenten in der ersten Verschachtelungsebene eines Arrays. In Amazon DocumentDB 4.0 durchläuft der Operator alle Ebenen, bevor er übereinstimmende Dokumente zurückgibt. Beispiel:

```
db.foo.insert(
[
    {a: {b: 5}},
    {a: {b: [5]}},
    {a: {b: [3, 7]}},
    {a: [{b: 5}]},
    {a: [{b: 3}, {b: 7}]},
    {a: [{b: [5]}]},
    {a: [{b: [3, 7]}]},
    {a: [[{b: 5}]]},
    {a: [[{b: 3}, {b: 7}]]},
    {a: [[{b: [5]}]]},
    {a: [[{b: [3, 7]}]]}
]);
// Elastic clusters
> db.foo.find({a: {$elemMatch: {b: {$elemMatch: {$lt: 6, $gt: 4}}}}}, {_id: 0})
```

```
{ "a" : [ { "b" : [ 5 ] } ] }

// Docdb 4.0: traverse more than one level deep
> db.foo.find({a: {$elemMatch: {b: {$elemMatch: {$lt: 6, $gt: 4}}}}}, {_id: 0})
{ "a" : [ { "b" : [ 5 ] } ] }
{ "a" : [ [ { "b" : [ 5 ] } ] ] }
```

- Die „$“ -Projektion in Amazon DocumentDB 4.0 gibt alle Dokumente mit allen Feldern zurück. Bei elastischen Clustern gibt der `find` Befehl mit einer „$“ -Projektion Dokumente zurück, die dem Abfrageparameter entsprechen und nur das Feld enthalten, das der „$“ -Projektion entspricht.
- In elastischen Clustern geben die `find` Befehle mit `$regex` und `$options` Abfrageparametern einen Fehler zurück: „Optionen können nicht sowohl in $regex als auch in $options festgelegt werden.“

- Bei elastischen Clustern wird `$indexOfCP` jetzt „-1“ zurückgegeben, wenn:
  - die Teilzeichenfolge wurde nicht in der `string expression` gefunden, oder
  - `start`ist eine Zahl größer als`end`, oder
  - `start`ist eine Zahl, die größer als die Bytelänge der Zeichenfolge ist.

  Gibt in Amazon DocumentDB 4.0 „0" `$indexOfCP` zurück, wenn die `start` Position eine Zahl ist, die größer als `end` oder die Bytelänge der Zeichenfolge ist.
- Bei elastischen Clustern geben Projektionsoperationen in `_id fields` z. B. Dokumente zurück`{"_id.nestedField" : 1}`, die nur das projizierte Feld enthalten. In der Zwischenzeit filtern in Amazon DocumentDB 4.0 Befehle zur Projektion verschachtelter Felder kein Dokument heraus.

Erste Schritte mit Amazon DocumentDB Elastic Clusters

In diesem Abschnitt „Erste Schritte“ erfahren Sie, wie Sie Ihren ersten elastischen Cluster erstellen und abfragen können.

Es gibt viele Möglichkeiten, eine Verbindung herzustellen und mit Amazon DocumentDB zu beginnen. Das folgende Verfahren ist der schnellste, einfachste und einfachste Weg für Benutzer, mit der Nutzung unserer leistungsstarken Dokumentendatenbank zu beginnen. In diesem Handbuch AWS CloudShell können Sie Ihren Amazon DocumentDB-Cluster direkt von der aus verbinden und abfragen. AWS-Managementkonsole Neukunden, die für das AWS kostenlose Kontingent in Frage

kommen, können Amazon DocumentDB CloudShell kostenlos nutzen. Wenn Ihre AWS CloudShell Umgebung oder Ihr Amazon DocumentDB Elastic Cluster Ressourcen nutzt, die über das kostenlose Kontingent hinausgehen, werden Ihnen die normalen AWS Tarife für diese Ressourcen berechnet. Mit diesem Handbuch können Sie in weniger als 5 Minuten mit Amazon DocumentDB loslegen.

Topics

- Voraussetzungen
- Schritt 1: Erstellen Sie einen elastischen Cluster
- Schritt 2: Stellen Sie eine Verbindung zu Ihrem Elastic Cluster her
- Schritt 3: Teilen Sie Ihre Sammlung, fügen Sie Daten ein und fragen Sie sie ab
- Schritt 4: Erkunden

Voraussetzungen

Bevor Sie Ihren ersten Amazon DocumentDB-Cluster erstellen, müssen Sie Folgendes tun:

Erstellen Sie ein Amazon Web Services (AWS) -Konto

Bevor Sie Amazon DocumentDB verwenden können, müssen Sie über ein Amazon Web Services (AWS) -Konto verfügen. Das AWS Konto ist kostenlos. Sie zahlen nur für die Services und Ressourcen, die Sie wirklich nutzen.

Wenn Sie noch keines haben AWS-Konto, führen Sie die folgenden Schritte aus, um eines zu erstellen.

Um sich für eine anzumelden AWS-Konto

1. Offen https://signin.aws.amazon.com/signup? request_type=registrieren.
2. Folgen Sie den Online-Anweisungen.

   Während der Anmeldung erhalten Sie einen Telefonanruf oder eine Textnachricht und müssen einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für eine registrieren AWS-Konto, wird eine erstellt. Root-Benutzer des AWS-Kontos Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Benutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

Richten Sie die erforderlichen AWS Identity and Access Management (IAM-) Berechtigungen ein.

Für den Zugriff auf die Verwaltung von Amazon DocumentDB-Ressourcen wie Clustern, Instances und Cluster-Parametergruppen sind Anmeldeinformationen erforderlich, mit denen Sie Ihre Anfragen authentifizieren AWS können. Weitere Informationen finden Sie unter Identitäts- und Zugriffsmanagement für Amazon DocumentDB.

1. Geben Sie in der Suchleiste von IAM ein **IAM** und wählen Sie es im Drop-down-Menü aus. AWS-Managementkonsole
2. Sobald Sie sich in der IAM-Konsole befinden, wählen Sie im Navigationsbereich Benutzer aus.
3. Wählen Sie Ihren Nutzernamen aus.
4. Wählen Sie Add permissions (Berechtigungen hinzufügen) aus.
5. Wählen Sie Richtlinien direkt anhängen aus.
6. Geben Sie es `AmazonDocDBElasticFullAccess` in die Suchleiste ein und wählen Sie es aus, sobald es in den Suchergebnissen erscheint.
7. Wählen Sie Weiter aus.
8. Wählen Sie Add permissions (Berechtigungen hinzufügen) aus.

> **Note**
>
> Ihr AWS Konto beinhaltet eine Standard-VPC in jeder Region. Wenn Sie sich für die Verwendung einer Amazon VPC entscheiden, führen Sie die Schritte im Thema Erstellen einer Amazon VPC im Amazon VPC-Benutzerhandbuch aus.

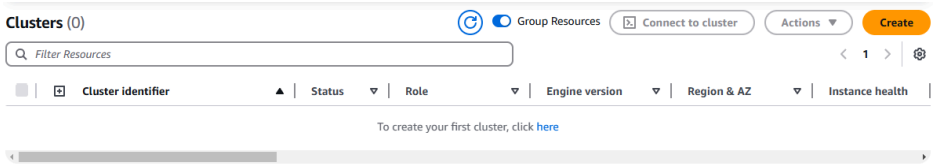
Schritt 1: Erstellen Sie einen elastischen Cluster

In diesem Abschnitt erklären wir, wie Sie einen brandneuen elastischen Cluster erstellen, indem Sie entweder die AWS-Managementkonsole oder AWS CLI die folgenden Anweisungen verwenden.

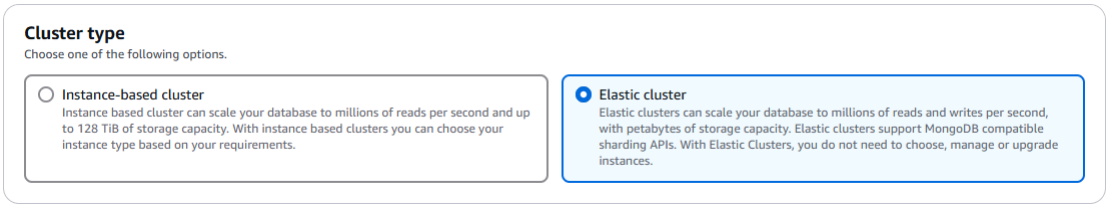
Using the AWS-Managementkonsole

So erstellen Sie eine elastische Cluster-Konfiguration mit AWS-Managementkonsole:

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter https://console.aws.amazon.com/docdb.
2. Wählen Sie in der Amazon DocumentDB Management Console unter Clusters die Option Create aus.



3. Wählen Sie auf der Seite „Amazon DocumentDB-Cluster erstellen“ im Abschnitt Cluster-Typ die Option Elastic Cluster aus.



4. Konfigurieren Sie im Abschnitt Konfiguration Folgendes:

   a. Geben Sie im Feld Clustername eine eindeutige Cluster-ID ein (gemäß den Benennungsanforderungen unter dem Feld).

   b. Geben Sie im Feld Anzahl der Shards die Anzahl der Shards ein, die Sie in Ihrem Cluster haben möchten. Die maximale Anzahl von Shards pro Cluster beträgt 32.

   > **Note**
   >
   > Für jeden Shard werden zwei Knoten bereitgestellt. Beide Knoten werden dieselbe Shard-Kapazität haben.

   c. Wählen Sie im Feld Anzahl der Shard-Instances die Anzahl der Replikat-Instances aus, die Sie jedem Shard zuordnen möchten. Die maximale Anzahl von Shard-Instances beträgt 16 in Schritten von 1. Alle Replikatinstanzen haben dieselbe Shard-Kapazität, wie im folgenden Feld definiert. Zu Testzwecken sollte der Standardwert 2 ausreichen.

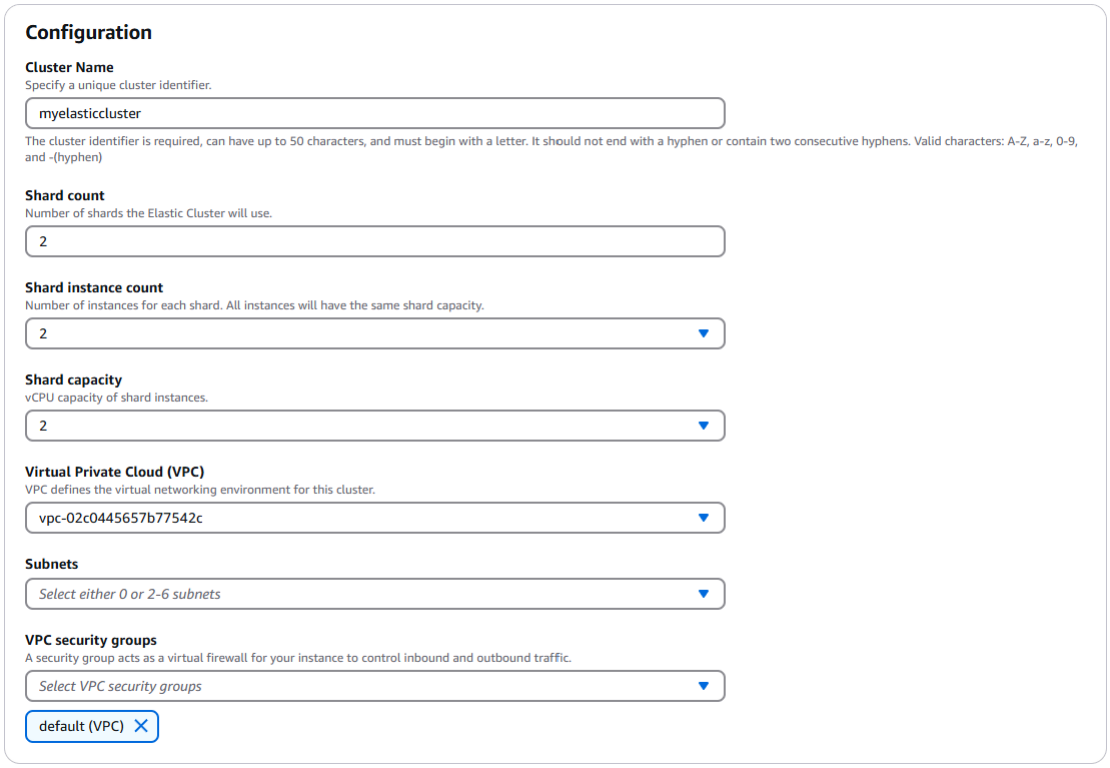
   > **Note**
   >
   > Die Anzahl der Replikatinstanzen gilt für alle Shards im Elastic Cluster. Ein Wert von 1 für die Anzahl der Shard-Instances bedeutet, dass es eine Writer-Instanz gibt und alle weiteren Instanzen Replikate sind, die für Lesevorgänge und zur

Verbesserung der Verfügbarkeit verwendet werden können. Zu Testzwecken sollte der Standardwert 2 ausreichen.

d. Wählen Sie im Feld Shard-Kapazität die Anzahl der virtuellen CPUs (vCPUs) aus, die Sie jeder Shard-Instance zuordnen möchten. Die maximale Anzahl von vCPUs pro Shard-Instance beträgt 64. Zulässige Werte sind 2, 4, 8, 16, 32, 64. Zu Testzwecken sollte der Standardwert 2 ausreichen.

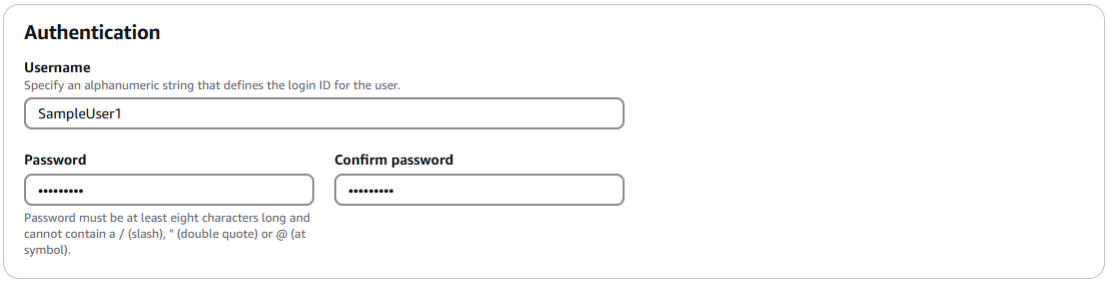
e. Wählen Sie im Feld Virtual Private Cloud (VPC) eine VPC aus der Dropdownliste aus.

f. Für Subnetze und VPC-Sicherheitsgruppen können Sie die Standardeinstellungen verwenden oder drei Subnetze Ihrer Wahl und bis zu drei VPC-Sicherheitsgruppen (mindestens eine) auswählen.



5. Geben Sie im Abschnitt Authentifizierung im Feld Benutzername eine Zeichenfolge ein, die den Anmeldenamen des Hauptbenutzers identifiziert.

   Geben Sie im Feld Passwort ein eindeutiges Passwort ein, das den Anweisungen entspricht, und bestätigen Sie es dann.



6. Behalten Sie im Abschnitt Verschlüsselung die Standardeinstellung (Standardschlüssel) bei.

   Optional können Sie einen von Ihnen erstellten AWS KMS key ARN eingeben. Weitere Informationen finden Sie unter Datenverschlüsselung im Ruhezustand für elastische Amazon DocumentDB-Cluster.

   > **Important**
   >
   > Die Verschlüsselung muss für elastische Cluster aktiviert sein.

7. Bearbeiten Sie im Abschnitt Backup die Felder entsprechend Ihren Backup-Anforderungen. Zu Testzwecken können Sie die Standardeinstellungen beibehalten.



   a. Aufbewahrungszeitraum für Backups — Wählen Sie in der Liste die Anzahl der Tage aus, für die automatische Backups dieses Clusters aufbewahrt werden sollen, bevor sie gelöscht werden.

   b. Backup-Fenster — Legen Sie die tägliche Uhrzeit und Dauer fest, während der Amazon DocumentDB Backups dieses Clusters erstellen soll.

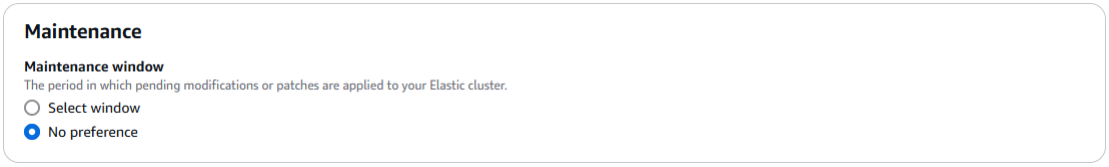
      i. Wählen Sie Fenster auswählen, wenn Sie den Zeitpunkt und die Dauer der Erstellung von Backups konfigurieren möchten.

         Startzeit — Wählen Sie in der ersten Liste die Startzeit (Stunde) (UTC) für den Start Ihrer automatischen Backups aus. Wählen Sie in der zweiten Liste die Minute für den Beginn der automatischen Backups aus.

Dauer — Wählen Sie in der Liste die Anzahl der Stunden aus, die der Erstellung automatischer Backups zugewiesen werden sollen.

ii. Wählen Sie Keine Präferenz, wenn Amazon DocumentDB den Zeitpunkt und die Dauer der Erstellung von Backups auswählen soll.

8. Wählen Sie im Abschnitt Wartung den Tag, die Uhrzeit und die Dauer aus, an denen Änderungen oder Patches auf Ihren Cluster angewendet werden. Zu Testzwecken können Sie die Standardeinstellungen beibehalten.



9. Wählen Sie Cluster erstellen.

Der elastische Cluster wird jetzt bereitgestellt. Es kann bis zu ein paar Minuten dauern, bis dieser Vorgang abgeschlossen ist. Sie können eine Verbindung zu Ihrem Cluster herstellen, wenn der Elastic Cluster-Status in der Cluster-Liste als Verfügbar angezeigt wird.

Using the AWS CLI

Verwenden Sie die `create-cluster` Operation mit den folgenden Parametern AWS CLI, um einen elastischen Cluster mit den folgenden Parametern zu erstellen:

- --cluster-name—Erforderlich. Der aktuelle Name des Elastic Scale Clusters, wie er bei der Erstellung eingegeben oder zuletzt geändert wurde.
- --shard-capacity—Erforderlich. Die Anzahl der vCPUs, die jedem Shard zugewiesen sind. Das Maximum ist 64. Zulässige Werte sind 2, 4, 8, 16, 32, 64.
- --shard-count—Erforderlich. Die Anzahl der Shards, die dem Cluster zugewiesen sind. Das Maximum ist 32.
- --shard-instance-count— Fakultativ. Die Anzahl der Replikatinstanzen, die für alle Shards in diesem Cluster gelten. Das Maximum ist 16.
- --admin-user-name—Erforderlich. Der dem Admin-Benutzer zugeordnete Benutzername.
- --admin-user-password—Erforderlich. Das dem Admin-Benutzer zugeordnete Passwort.
- --auth-type—Erforderlich. Der Authentifizierungstyp, der verwendet wird, um zu bestimmen, wo das für den Zugriff auf den Elastic Cluster verwendete Passwort abgerufen werden soll. Gültige Typen sind `PLAIN_TEXT` oder`SECRET_ARN`.

- --vpc-security-group-ids— Fakultativ. Konfigurieren Sie eine Liste von EC2-VPC-Sicherheitsgruppen, die diesem Cluster zugeordnet werden sollen.
- --preferred-maintenance-window— Fakultativ. Konfigurieren Sie den wöchentlichen Zeitraum, in dem Systemwartungen stattfinden können, in koordinierter Weltzeit (UTC).

  Das Format ist:`ddd:hh24:mi-ddd:hh24:mi`. Gültige Tage (ddd): Mo, Di, Mi, Do, Fr, Sa, So

  Die Standardeinstellung ist ein 30-minütiges Fenster, das nach dem Zufallsprinzip aus einem 8-Stunden-Zeitblock für jede Amazon Web Services-Region ausgewählt wird und an einem zufälligen Wochentag stattfindet.

  Zeitfenster von mindestens 30 Minuten.
- --kms-key-id— Fakultativ. Konfigurieren Sie die KMS-Schlüssel-ID für einen verschlüsselten Cluster.

  Die KMS-Schlüssel-ID ist der Amazon-Ressourcenname (ARN) für den AWS KMS Verschlüsselungsschlüssel. Wenn Sie einen Cluster mit demselben Amazon Web Services-Konto erstellen, dem der KMS-Verschlüsselungsschlüssel gehört, der zum Verschlüsseln des neuen Clusters verwendet wird, können Sie den KMS-Schlüsselalias anstelle des ARN für den KMS-Verschlüsselungsschlüssel verwenden.

  Wenn in diesem Feld kein Verschlüsselungsschlüssel angegeben ist KmsKeyId und der `StorageEncrypted` Parameter wahr ist, verwendet Amazon DocumentDB Ihren Standard-Verschlüsselungsschlüssel.
- --preferred-backup-window— Fakultativ. Der bevorzugte tägliche Zeitraum, in dem automatische Backups erstellt werden. Die Standardeinstellung ist ein 30-minütiges Fenster, das nach dem Zufallsprinzip aus einem Zeitblock von jeweils AWS-Region 8 Stunden ausgewählt wird.
- --backup-retention-period— Fakultativ. Die Anzahl von Tagen, über die automatische Backups aufbewahrt werden. Der Standardwert ist 1.
- --storage-encrypted— Fakultativ. Konfiguriert, ob der Cluster verschlüsselt oder nicht verschlüsselt ist.

  `--no-storage-encrypted`gibt an, dass der Cluster nicht verschlüsselt ist.
- --subnet-ids— Fakultativ. Konfigurieren Sie Netzwerk-Subnetz-IDs.

Ersetzen Sie im folgenden Beispiel jede *user input placeholder* durch Ihre eigenen Informationen.

> **Note**
>
> Die folgenden Beispiele beinhalten die Erstellung eines bestimmten KMS-Schlüssels. Um den Standard-KMS-Schlüssel zu verwenden, geben Sie den `--kms-key-id` Parameter nicht an.

Für Linux, macOS oder Unix:

```
aws docdb-elastic create-cluster \
     --cluster-name sample-cluster-123 \
     --shard-capacity 8 \
     --shard-count 4 \
     --shard-instance-count 3 \
     --auth-type PLAIN_TEXT \
     --admin-user-name testadmin \
     --admin-user-password testPassword \
     --vpc-security-group-ids ec-65f40350 \
     --kms-key-id arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2 \
     --subnet-ids subnet-9253c6a3, subnet-9f1b5af9 \
     --preferred-backup-window 18:00-18:30 \
     --backup-retention-period 7
```

Für Windows:

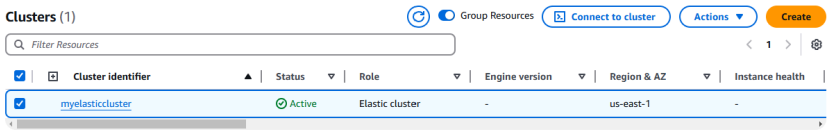
```
aws docdb-elastic create-cluster ^
     --cluster-name sample-cluster-123 ^
     --shard-capacity 8 ^
     --shard-count 4 ^
     --shard-instance-count 3 ^
     --auth-type PLAIN_TEXT ^
     --admin-user-name testadmin ^
     --admin-user-password testPassword ^
     --vpc-security-group-ids ec-65f40350 ^
     --kms-key-id arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2 ^
     --subnet-ids subnet-9253c6a3, subnet-9f1b5af9 \
```

```
    --preferred-backup-window 18:00-18:30 \
    --backup-retention-period 7
```

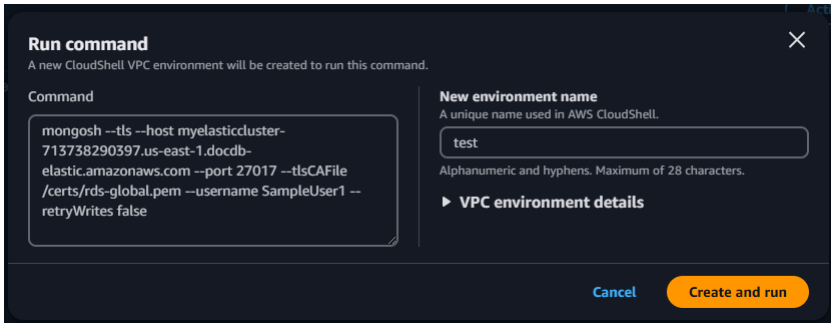
Schritt 2: Stellen Sie eine Verbindung zu Ihrem Elastic Cluster her

Stellen Sie mithilfe AWS CloudShell von eine Verbindung zu Ihrem Amazon DocumentDB Elastic Cluster her.

1. Suchen Sie in der Amazon DocumentDB-Managementkonsole unter Clusters nach dem elastischen Cluster, den Sie erstellt haben. Wählen Sie Ihren Cluster aus, indem Sie das Kontrollkästchen daneben aktivieren.



2. Wählen Sie Mit Cluster verbinden (dies befindet sich neben dem Dropdownmenü Aktionen). Diese Schaltfläche ist erst aktiviert, nachdem Sie auf das Kontrollkästchen neben Ihrem Cluster geklickt haben und der Status des Clusters als Verfügbar angezeigt wird. Der Bildschirm „Befehl CloudShell ausführen“ wird angezeigt.
3. Geben Sie im Feld Neuer Umgebungsname einen eindeutigen Namen ein, z. B. „Test“, und wählen Sie Erstellen und ausführen aus. Die VPC-Umgebungsdetails werden automatisch für Ihre Amazon DocumentDB-Datenbank konfiguriert.



4. Wenn Sie dazu aufgefordert werden, geben Sie das Passwort ein, das Sie in Schritt 1: Erstellen Sie einen Amazon DocumentDB Elastic Cluster (Unterschritt 5) erstellt haben.



Nachdem Sie Ihr Passwort eingegeben haben und Ihre Eingabeaufforderung erscheint`direct: mongos] <env-name>>`, sind Sie erfolgreich mit Ihrem Amazon DocumentDB-Cluster verbunden

> **Note**
>
> Informationen zur Fehlerbehebung finden Sie unter Problembehandlung bei Amazon DocumentDB.

Schritt 3: Teilen Sie Ihre Sammlung, fügen Sie Daten ein und fragen Sie sie ab

Elastische Cluster bieten Unterstützung für Sharding in Amazon DocumentDB. Jetzt, da Sie mit Ihrem Cluster verbunden sind, können Sie den Cluster teilen, Daten einfügen und einige Abfragen ausführen.

1. Um eine Sammlung zu teilen, geben Sie Folgendes ein:

```
sh.shardCollection("db.Employee1" , { "Employeeid" : "hashed" })
```

2. Um ein einzelnes Dokument einzufügen, geben Sie Folgendes ein:

```
db.Employee1.insertOne({"Employeeid":1, "Name":"Joe", "LastName": "Bruin", "level":
 1 })
```

Die folgende Ausgabe wird angezeigt:

```
WriteResult({ "nInserted" : 1 })
```

3. Um das von Ihnen geschriebene Dokument zu lesen, geben Sie den `findOne()` folgenden Befehl ein (es wird ein einzelnes Dokument zurückgegeben):

```
db.Employee1.findOne()
```

Die folgende Ausgabe wird angezeigt:

```
{
"_id" : ObjectId("61f344e0594fe1a1685a8151"),
"EmployeeID" : 1,
"Name" : "Joe",
"LastName" : "Bruin",
"level" : 1
}
```

4. Um ein paar weitere Abfragen durchzuführen, ziehen Sie einen Anwendungsfall für ein Spieleprofil in Betracht. Fügen Sie zunächst einige Einträge in eine Sammlung mit dem Titel „Mitarbeiter“ ein. Geben Sie Folgendes ein:

```
db.profiles.insertMany([ { "_id": 1, "name": "Matt", "status": "active", "level":
 12, "score": 202 },
     { "_id": 2, "name": "Frank", "status": "inactive", "level": 2, "score": 9 },
     { "_id": 3, "name": "Karen", "status": "active", "level": 7, "score": 87 },
     { "_id": 4, "name": "Katie", "status": "active", "level": 3, "score": 27 }
])
```

Die folgende Ausgabe wird angezeigt:

```
{ acknowledged: true,
     insertedIds: {
        '0': ObjectId('679d02cd6b5a0581be78bcbd'),
        '1': ObjectId('679d02cd6b5a0581be78bcbe'),
        '2': ObjectId('679d02cd6b5a0581be78bcbf'),
        '3': ObjectId('679d02cd6b5a0581be78bcc0')
    }
}
```

5. Um alle Dokumente in der Profilsammlung zurückzugeben, geben Sie den Befehl `find` () ein:

```
db.Employee.find()
```

Die Daten, die Sie in Schritt 4 eingegeben haben, werden angezeigt.

6. Um ein einzelnes Dokument abzufragen, fügen Sie einen Filter hinzu (zum Beispiel: „Katie“). Geben Sie Folgendes ein:

```
db.Employee.find({name: "Katie"})
```

Die folgende Ausgabe wird angezeigt:

```
[

   {

     _id: ObjectId('679d02cd6b5a0581be78bcc0'),

     Employeeid: 4,

     name: 'Katie',

     lastname: 'Schaper',

     level: 3

   }

]
```

7. Um ein Profil zu finden und es zu ändern, geben Sie den `findAndModify` Befehl ein. In diesem Beispiel erhält der Mitarbeiter „Matt“ die höhere Stufe „14":

```
db.Employee.findAndModify({
    query: { "Employeeid" : 1, "name" : "Matt"},
    update: { "Employeeid" : 1, "name" : "Matt", "lastname" : "Winkle", "level" :
 14 }
})
```

Die folgende Ausgabe wird angezeigt (beachten Sie, dass sich das Level noch nicht geändert hat):

```
{
```

```
   _id: ObjectId('679d02cd6b5a0581be78bcbd'),

   Employeeid: 1,

   name: 'Matt',

   lastname: 'Winkle',

   level: 12

}
```

8. Geben Sie die folgende Abfrage ein, um den Levelanstieg zu überprüfen:

```
db.Employee.find({name: "Matt"})
```

Die folgende Ausgabe wird angezeigt:

```
[
   {

   _id: ObjectId('679d02cd6b5a0581be78bcbd'),

   Employeeid: 1,

   name: 'Matt',

   lastname: 'Winkle',

   level: 14

   }
]
```

Schritt 4: Erkunden

Herzlichen Glückwunsch! Sie haben das Verfahren „Erste Schritte“ für Amazon DocumentDB Elastic Clusters erfolgreich abgeschlossen.

Was kommt als Nächstes? Erfahren Sie, wie Sie diese Datenbank mit einigen ihrer beliebten Funktionen voll ausschöpfen können:

- Bewährte Methoden für Amazon DocumentDB Elastic Cluster
- Elastische Amazon DocumentDB-Cluster verwalten

> **Note**
>
> Für den elastischen Cluster, den Sie im Rahmen dieser ersten Schritte erstellt haben, fallen weiterhin Kosten an, sofern Sie ihn nicht löschen. Eine Anleitung finden Sie unter Löschen eines elastischen Clusters.

Bewährte Methoden für Amazon DocumentDB Elastic Cluster

Erfahren Sie mehr über bewährte Methoden für die Arbeit mit elastischen Amazon DocumentDB-Clustern. Alle bewährten Methoden für instanzbasierte Amazon DocumentDB-Cluster gelten auch für elastische Cluster. Dieser Abschnitt wird fortlaufend aktualisiert, wenn neue bewährte Methoden identifiziert werden.

Topics

- Auswahl von Shard-Schlüsseln
- Verbindungsverwaltung
- Ungeteilte Sammlungen
- Skalierung elastischer Cluster
- Überwachung elastischer Cluster

Auswahl von Shard-Schlüsseln

In der folgenden Liste werden Richtlinien für die Erstellung von Shard-Schlüsseln beschrieben.

- Verwenden Sie einen gleichmäßig verteilten Hash-Schlüssel, um Ihre Daten auf alle Shards in Ihrem Cluster zu verteilen (vermeiden Sie Hotkeys).
- Verwenden Sie Ihren Shard-Schlüssel in allen read/update /delete-Anfragen, um Scatter Gather-Abfragen zu vermeiden.
- Vermeiden Sie verschachtelte Shard-Schlüssel, wenn Sie /delete-Operationen ausführen. read/update

- Wenn Sie Batch-Operationen ausführen, setzen Sie `ordered` den Wert auf false, damit alle Shards parallel ausgeführt werden können und die Latenzen verbessert werden.

Verbindungsverwaltung

Die folgende Liste beschreibt Richtlinien für die Verwaltung Ihrer Verbindungen zu Ihrer Datenbank.

- Überwachen Sie die Anzahl Ihrer Verbindungen und wie oft neue Verbindungen geöffnet und geschlossen werden.
- Verteilen Sie Ihre Verbindungen auf alle Subnetze in der Konfiguration Ihrer Anwendung. Wenn Ihr Cluster in mehreren Subnetzen konfiguriert ist, Sie aber nur eine Teilmenge der Subnetze verwenden, kann es zu Engpässen bei Ihrer maximalen Anzahl von Verbindungen kommen.

Ungeteilte Sammlungen

Im Folgenden wird eine Richtlinie für Sammlungen ohne gemeinsame Nutzung beschrieben.

- Wenn Sie mit Sammlungen ohne Sharded arbeiten, sollten Sie zur Lastverteilung versuchen, häufig genutzte Sammlungen ohne Sharded in verschiedenen Datenbanken zu speichern. Elastische Amazon DocumentDB-Cluster platzieren Datenbanken auf verschiedenen Shards und platzieren Sammlungen, die nicht gemeinsam genutzt wurden, für dieselbe Datenbank auf demselben Shard.

Skalierung elastischer Cluster

In der folgenden Liste werden Richtlinien für die Skalierung Ihrer elastischen Cluster beschrieben.

- Skalierungsvorgänge können kurzzeitig zu zeitweiligen Datenbank- und Netzwerkfehlern führen. Vermeiden Sie nach Möglichkeit die Skalierung zu Spitzenzeiten. Versuchen Sie, während Wartungszeitfenstern zu skalieren.
- Das Hoch- und Herunterskalieren der Shard-Kapazität (Änderung der vCPU-Anzahl pro Shard) zur Erhöhung der Rechenleistung wird dem Erhöhen oder Verringern der Shard-Anzahl vorgezogen, da dies schneller ist und die Dauer intermittierender Datenbank- und Netzwerkfehler kürzer ist.
- Wenn Sie mit einem Wachstum rechnen, ziehen Sie es vor, die Anzahl der Shards zu erhöhen, anstatt die Shard-Kapazität zu skalieren. Auf diese Weise können Sie Ihren Cluster skalieren, indem Sie die Shard-Kapazität für Szenarien erhöhen, in denen Sie schnell skalieren müssen.

- Überwachen Sie Ihre clientseitigen Wiederholungsrichtlinien und versuchen Sie es erneut mit exponentiellem Backoff und Jitter, um eine Überlastung Ihrer Datenbank zu vermeiden, wenn bei der Skalierung Fehler auftreten.
- Das Ändern der Anzahl der Shards kann einige Zeit in Anspruch nehmen, da Daten sicher von einem Shard zum anderen verschoben werden müssen. Um das Fehlerrisiko zu minimieren, unterbrechen Sie den Cluster-Status nicht und nehmen Sie keine anderen Änderungen am Cluster-Status vor, während die Änderung im Gange ist.

Überwachung elastischer Cluster

In der folgenden Liste werden Richtlinien für die Überwachung Ihrer Elastic Cluster beschrieben.

- Verfolgen Sie das Verhältnis zwischen Spitze und Durchschnitt Ihrer Kennzahlen pro Shard, um festzustellen, ob Sie ungleichmäßigen Traffic generieren (einen Hotspot haben). key/hot Die wichtigsten Kennzahlen zur Erfassung des Spitzenwerts im Vergleich zum Durchschnittswert sind:
  - `PrimaryInstanceCPUUtilization`
    - Dies kann auf der Ebene pro Shard überwacht werden.
    - Auf Cluster-Ebene können Sie den Durchschnitt bis auf p99 Skew überwachen.
  - `PrimaryInstanceFreeableMemory`
    - Dies kann auf der Ebene pro Shard überwacht werden.
    - Auf Cluster-Ebene können Sie den Durchschnitt bis auf p99 Skew überwachen.
  - `DatabaseCursorsMax`
    - Dies sollte auf der Ebene pro Shard überwacht werden, um den Skew zu ermitteln.
  - `Documents-Inserted/Updated/Returned/Deleted`
    - Dies sollte auf der Ebene pro Shard überwacht werden, um den Skew zu ermitteln.

Elastische Amazon DocumentDB-Cluster verwalten

Um einen elastischen Amazon DocumentDB-Cluster zu verwalten, benötigen Sie eine IAM-Richtlinie mit den entsprechenden Amazon DocumentDB-Steuerplane-Berechtigungen. Mit diesen Berechtigungen können Sie Cluster erstellen, ändern und löschen. Die DocumentDBFullAccess Amazon-Richtlinie bietet alle erforderlichen Berechtigungen für die Verwaltung eines Amazon DocumentDB Elastic Clusters.

Die folgenden Themen zeigen, wie Sie bei der Arbeit mit elastischen Amazon DocumentDB-Clustern verschiedene Aufgaben ausführen.

Topics

- Elastische Cluster-Konfigurationen ändern
- Überwachung eines elastischen Clusters
- Löschen eines elastischen Clusters
- Verwaltung von Elastic Cluster-Snapshots
- Stoppen und Starten eines elastischen Amazon DocumentDB-Clusters
- Wartung der elastischen Amazon DocumentDB-Cluster

Elastische Cluster-Konfigurationen ändern

In diesem Abschnitt erklären wir, wie Sie den Elastic Cluster ändern, indem Sie entweder die AWS-Managementkonsole oder AWS CLI die folgenden Anweisungen verwenden.

Das Ändern des Clusters wird hauptsächlich zur Skalierung von Shards verwendet, indem die Anzahl and/or der Shards erhöht oder verringert wird.

Using the AWS-Managementkonsole

So ändern Sie eine Elastic Cluster-Konfiguration mit: AWS-Managementkonsole

1. Melden Sie sich bei der Amazon DocumentDB-Konsole an AWS-Managementkonsole und öffnen Sie sie.
2. Klicken Sie im Navigationsbereich auf Cluster.

> **Tip**
>
> Wenn Sie den Navigationsbereich nicht auf der linken Seite Ihres Bildschirms sehen, wählen Sie das Menüsymbol in der oberen linken Ecke des Navigationsbereichs.

3. Wählen Sie in der Spalte Cluster-ID den Namen des Clusters aus, den Sie ändern möchten.
4. Wählen Sie Ändern aus.
5. Bearbeiten Sie die Felder, die Sie ändern möchten, und wählen Sie dann Cluster ändern aus.

Configuration

Cluster identifier

SampleCluster

Shard count

Number of shards the Elastic Cluster will use.

2

Shard instance count

Number of instances for each shard. All instances will have the same shard capacity.

2

Shard capacity

vCPU capacity of each shard.

2

Maintenance

Maintenance window

The period in which pending modifications or patches are applied to your Elastic cluster.

- ○ Select window
- ◉ No preference

Authentication

Username

SampleUser

New password

Confirm new password

Password must be at least eight characters long and cannot contain a / (slash), " (double quote) or @ (at symbol).

Network settings

Subnets

Select either 0 or 2-6 subnets

subnet-0b2962f92a0f5a8fb ✕ subnet-08c6d849efd4dfe96 ✕

VPC security groups

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic.

Select VPC security groups

sg-021d234a0a3a2c2fe ✕ sg-0824c1992d6009b52 ✕

> **Note**
>
> Sie können auch auf das Dialogfeld „Cluster ändern“ zugreifen, indem Sie auf die Seite „Cluster“ gehen, das Kästchen neben Ihrem Cluster markieren, „Aktionen“ und dann „Ändern“ auswählen.

Using the AWS CLI

Um eine Elastic-Cluster-Konfiguration mithilfe von zu ändern AWS CLI, verwenden Sie den `update-cluster` Vorgang mit den folgenden Parametern:

- **`--cluster-arn`**—Erforderlich. Die ARN-ID des Clusters, den Sie ändern möchten.
- **`--shard-capacity`**— Optional. Die Anzahl der vCPUs, die jedem Shard zugewiesen sind. Das Maximum ist 64. Zulässige Werte sind 2, 4, 8, 16, 32, 64.
- **`--shard-count`**— Fakultativ. Die Anzahl der Shards, die dem Cluster zugewiesen sind. Das Maximum ist 32.
- **`--shard-instance`**-count — optional. Die Anzahl der Replikatinstanzen, die für alle Shards in diesem Cluster gelten. Das Maximum ist 16.
- **`--auth-type`**— Fakultativ. Der Authentifizierungstyp, der verwendet wird, um zu bestimmen, wo das für den Zugriff auf den Elastic Cluster verwendete Passwort abgerufen werden soll. Gültige Typen sind `PLAIN_TEXT` oder`SECRET_ARN`.
- **`--admin-user-password`**— Fakultativ. Das dem Admin-Benutzer zugeordnete Passwort.
- **`--vpc-security-group-ids`**— Fakultativ. Konfigurieren Sie eine Liste von Amazon EC2- und Amazon Virtual Private Cloud (VPC) -Sicherheitsgruppen, die diesem Cluster zugeordnet werden sollen.
- **`--preferred-maintenance-window`**— Fakultativ. Konfigurieren Sie den wöchentlichen Zeitraum, in dem Systemwartungen stattfinden können, in koordinierter Weltzeit (UTC)

  Das Format ist:`ddd:hh24:mi-ddd:hh24:mi`. Gültige Tage (ddd): Mo, Di, Mi, Do, Fr, Sa, So

  Die Standardeinstellung ist ein 30-minütiges Fenster, das nach dem Zufallsprinzip aus einem 8-Stunden-Zeitblock für jede Amazon Web Services-Region ausgewählt wird und an einem zufälligen Wochentag stattfindet.

  Zeitfenster von mindestens 30 Minuten.

- **`--subnet-ids`**— Fakultativ. Konfigurieren Sie Netzwerk-Subnetz-IDs.

Ersetzen Sie im folgenden Beispiel jede *`user input placeholder`* durch Ihre eigenen Informationen.

Für Linux, macOS oder Unix:

```
aws docdb-elastic update-cluster \
    --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2 \
    --shard-capacity 8 \
    --shard-count 4 \
    --shard-instance-count 3 \
    --admin-user-password testPassword \
    --vpc-security-group-ids ec-65f40350 \
    --subnet-ids subnet-9253c6a3, subnet-9f1b5af9
```

Für Windows:

```
aws docdb-elastic update-cluster ^
    --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2 ^
    --shard-capacity 8 ^
    --shard-count 4 ^
    --shard-instance-count 3 ^
    --admin-user-password testPassword ^
    --vpc-security-group-ids ec-65f40350 ^
    --subnet-ids subnet-9253c6a3, subnet-9f1b5af9
```

Informationen zum Überwachen des Status des Elastic Clusters nach Ihrer Änderung finden Sie unter Überwachen eines elastischen Clusters.

Überwachung eines elastischen Clusters

In diesem Abschnitt erklären wir, wie Sie Ihren Elastic Cluster überwachen, indem Sie entweder die AWS-Managementkonsole oder AWS CLI die folgenden Anweisungen verwenden.

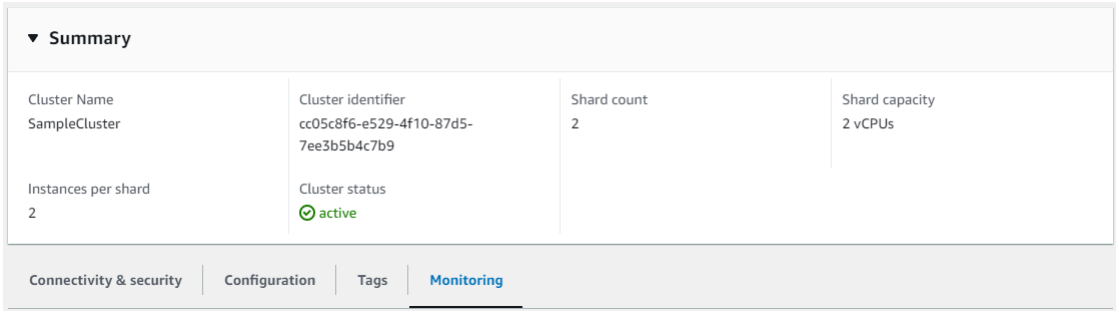
Using the AWS-Managementkonsole

So überwachen Sie eine Elastic Cluster-Konfiguration mit AWS-Managementkonsole:

1. Melden Sie sich bei der Amazon DocumentDB-Konsole an AWS-Managementkonsole und öffnen Sie sie.
2. Klicken Sie im Navigationsbereich auf Cluster.

> **Tip**
>
> Wenn Sie den Navigationsbereich nicht auf der linken Seite Ihres Bildschirms sehen, wählen Sie das Menüsymbol in der oberen linken Ecke des Navigationsbereichs.

3. Wählen Sie in der Spalte Cluster-ID den Namen des Clusters aus, den Sie überwachen möchten.
4. Wählen Sie die Registerkarte Überwachung.



Eine Reihe von Diagrammen von Amazon CloudWatch wird für die folgenden Überwachungskategorien angezeigt:

- Auslastung der Ressourcen
- Durchsatz
- Operationen
- System (System)

Sie können auch CloudWatch über die auf Amazon zugreifen AWS-Managementkonsole , um Ihre eigene Überwachungsumgebung für Ihre Elastic Cluster einzurichten.

Using the AWS CLI

Um eine bestimmte Elastic-Cluster-Konfiguration mithilfe von zu überwachen AWS CLI, verwenden Sie den `get-cluster` Vorgang mit den folgenden Parametern:

- **`--cluster-arn`**—Erforderlich. Die ARN-ID des Clusters, für den Sie Informationen benötigen.

Ersetzen Sie im folgenden Beispiel jedes *`user input placeholder`* durch Ihre eigenen Informationen.

Für Linux, macOS oder Unix:

```
aws docdb-elastic get-cluster \
    --cluster-arn arn:aws:docdb-elastic:us-west-2:123456789012:cluster:/68ffcdf8-e3af-40a3-91e4-24736f2dacc9
```

Für Windows:

```
aws docdb-elastic get-cluster ^
    --cluster-arn arn:aws:docdb:-elastic:us-west-2:123456789012:cluster:/68ffcdf8-e3af-40a3-91e4-24736f2dacc9
```

Die Ausgabe dieses Vorgangs sieht in etwa wie folgt aus:

```
"cluster": {
        ...
        "clusterArn": "arn:aws:docdb-elastic:us-
west-2:123456789012:cluster:/68ffcdf8-e3af-40a3-91e4-24736f2dacc9",
        "clusterEndpoint": "stretch-11-477568257630.us-east-1.docdb-
elastic.amazonaws.com",
        "readerEndpoint": "stretch-11-477568257630-ro.us-east-1.docdb-
elastic.amazonaws.com",
        "clusterName": "stretch-11",
        "shardCapacity": 2,
        "shardCount": 3,
        "shardInstanceCount: 5,
        "status": "ACTIVE",
        ...
 }
```

Weitere Informationen finden Sie `DescribeClusterSnapshot` in der Amazon DocumentDB Resource Management API-Referenz.

Um die Details aller elastischen Cluster mithilfe von anzuzeigen AWS CLI, verwenden Sie die `list-clusters` Operation mit den folgenden Parametern:

- **`--next-token`**— Fakultativ. Wenn die Anzahl der ausgegebenen Elemente (`--max-results`) geringer als die Gesamtanzahl der Elemente ist, die von den zugrunde liegenden

API-Aufrufen zurückgeliefert werden, enthält die Ausgabe ein `NextToken`. Dieses können Sie in einem anschließenden Befehl zum Abrufen der nächsten Gruppe von Elementen übergeben.

- **`--max-results`**— Fakultativ. Die Gesamtzahl der Elemente, die in der Ausgabe des Befehls zurückgegeben werden sollen. Wenn mehr Ergebnisse als der angegebene `max-results` Wert vorhanden sind, ist ein Paginierungstoken (`next-token`) in der Antwort enthalten, sodass die verbleibenden Ergebnisse abgerufen werden können.
  - Standard: 100
  - Mindestens 20, maximal 100

Ersetzen Sie im folgenden Beispiel jedes *`user input placeholder`* durch Ihre eigenen Informationen.

Für Linux, macOS oder Unix:

```
aws docdb-elastic list-clusters \
    --next-token eyJNYXJrZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxfQ== \
    --max-results 2
```

Für Windows:

```
aws docdb-elastic list-clusters ^
    --next-token eyJNYXJrZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxfQ== ^
    --max-results 2
```

Die Ausgabe dieses Vorgangs sieht in etwa wie folgt aus:

```
{
   "Clusters": [
      {
         "ClusterIdentifier":"mycluster-1",
         "ClusterArn":"arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-
cluster"
         "Status":"available",
         "ClusterEndpoint":"sample-cluster.sharded-cluster-corcjozrlsfc.us-
west-2.docdb.amazonaws.com"
       }
       {
         "ClusterIdentifier":"mycluster-2",
         "ClusterArn":"arn:aws:docdb:us-west-2:987654321098:sharded-cluster:sample-
cluster"
```

```
         "Status":"available",
         "ClusterEndpoint":"sample-cluster2.sharded-cluster-corcjozrlsfc.us-
west-2.docdb.amazonaws.com"
       }
   ]
}
```

Löschen eines elastischen Clusters

In diesem Abschnitt erklären wir, wie Sie einen elastischen Cluster löschen, indem Sie entweder die AWS-Managementkonsole oder AWS CLI die folgenden Anweisungen verwenden.

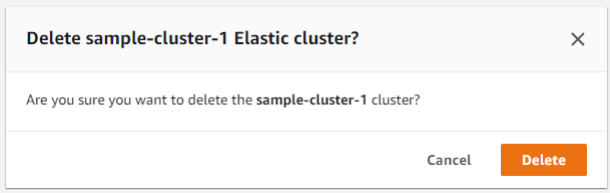
Using the AWS-Managementkonsole

So löschen Sie eine Elastic Cluster-Konfiguration mit AWS-Managementkonsole:

1. Melden Sie sich bei der Amazon DocumentDB-Konsole an AWS-Managementkonsole und öffnen Sie sie.
2. Klicken Sie im Navigationsbereich auf Cluster.

> **Tip**
>
> Wenn Sie den Navigationsbereich nicht auf der linken Seite Ihres Bildschirms sehen, wählen Sie das Menüsymbol in der oberen linken Ecke des Navigationsbereichs.

3. Markieren Sie in der Cluster-Listentabelle das Kontrollkästchen links neben dem Clusternamen, den Sie löschen möchten, und wählen Sie dann Aktionen aus. Wählen Sie im Dropdown-Menü Löschen aus.
4. Im elastischen Cluster „Clustername“ löschen? Wählen Sie im Dialogfeld Löschen aus.



Es dauert einige Minuten, bis der Cluster gelöscht ist. Informationen zur Überwachung des Status des Clusters finden Sie unter Überwachung des Status eines Amazon DocumentDB-Clusters.

Using the AWS CLI

Um einen elastischen Cluster mithilfe von zu löschen AWS CLI, verwenden Sie den `delete-cluster` Vorgang mit den folgenden Parametern:

- **`--cluster-arn`**—Erforderlich. Die ARN-ID des Clusters, den Sie löschen möchten.
- **`--no-skip-final-backup`**— Optional. Wenn Sie eine endgültige Sicherung wünschen, müssen Sie diesen Parameter mit einem Namen für die endgültige Sicherung angeben. Sie müssen entweder `--final-backup-identifier` oder `--skip-final-backup` angeben.
- **`--skip-final-backup`**— Fakultativ. Verwenden Sie diesen Parameter nur, wenn Sie vor dem Löschen Ihres Clusters kein endgültiges Backup erstellen möchten. Standardmäßig wird ein letzter Snapshot erstellt.

In den folgenden AWS CLI Codebeispielen wird ein Cluster mit einem ARN von arn:aws:docdb:us-west- 2:123456789012:sharded-cluster:sample-cluster mit einer endgültigen Sicherung gelöscht.

*`user input placeholder`*Ersetzen Sie im folgenden Beispiel jedes durch Ihre eigenen Informationen.

Für Linux, macOS oder Unix:

```
aws docdb-elastic delete-cluster \
    --cluster-arn arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-cluster \
    --no-skip-final-backup \
    --final-backup-identifier finalArnBU-arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-cluster
```

Für Windows:

```
aws docdb-elastic delete-cluster ^
    --cluster-arn arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-cluster ^
    --no-skip-final-backup ^
    --final-backup-identifier finalArnBU-arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-cluster
```

In den folgenden AWS CLI Codebeispielen wird ein Cluster mit dem ARN arn:aws:docdb:us-west- 2:123456789012:sharded-cluster:sample-cluster gelöscht, ohne ein endgültiges Backup zu erstellen.

*user input placeholder*Ersetzen Sie im folgenden Beispiel jedes durch Ihre eigenen Informationen.

Für Linux, macOS oder Unix:

```
aws docdb-elastic delete-cluster \
    --cluster-arn arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-cluster \
    --skip-final-backup \
```

Für Windows:

```
aws docdb-elastic delete-cluster ^
    --cluster-arn arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-cluster ^
    --skip-final-backup ^
```

Die Ausgabe des `delete-cluster` Vorgangs ist eine Anzeige des Clusters, das Sie löschen.

Es dauert einige Minuten, bis der Cluster gelöscht ist. Informationen zur Überwachung des Status des Clusters finden Sie unter Überwachung des Status eines Amazon DocumentDB-Clusters.

Verwaltung von Elastic Cluster-Snapshots

Manuelle Snapshots können erstellt werden, nachdem ein Elastic Cluster erstellt wurde. Automatische Backups werden in dem Moment erstellt, in dem der Elastic Cluster-Snapshot erstellt wird.

> **Note**
>
> Ihr Elastic Cluster muss sich in dem `Available` Zustand befinden, in dem ein manueller Snapshot erstellt werden kann.

In diesem Abschnitt wird erklärt, wie Sie Elastic Cluster-Snapshots erstellen, anzeigen, aus ihnen wiederherstellen und löschen können.

Die folgenden Themen zeigen, wie Sie bei der Arbeit mit Amazon DocumentDB Elastic Cluster-Snapshots verschiedene Aufgaben ausführen.

Topics

- Einen manuellen Elastic Cluster-Snapshot erstellen
- Einen Elastic Cluster-Snapshot anzeigen
- Wiederherstellen eines elastischen Clusters aus einem Snapshot
- Einen Elastic Cluster-Snapshot kopieren
- Löschen eines elastischen Cluster-Snapshots
- Verwaltung eines automatischen Backups eines Elastic Cluster-Snapshots

Einen manuellen Elastic Cluster-Snapshot erstellen

In diesem Abschnitt erklären wir, wie Sie einen manuellen Elastic-Cluster-Snapshot erstellen, indem Sie entweder die AWS-Managementkonsole oder AWS CLI die folgenden Anweisungen verwenden.

Using the AWS-Managementkonsole

So erstellen Sie einen manuellen Elastic-Cluster-Snapshot mit dem AWS-Managementkonsole:

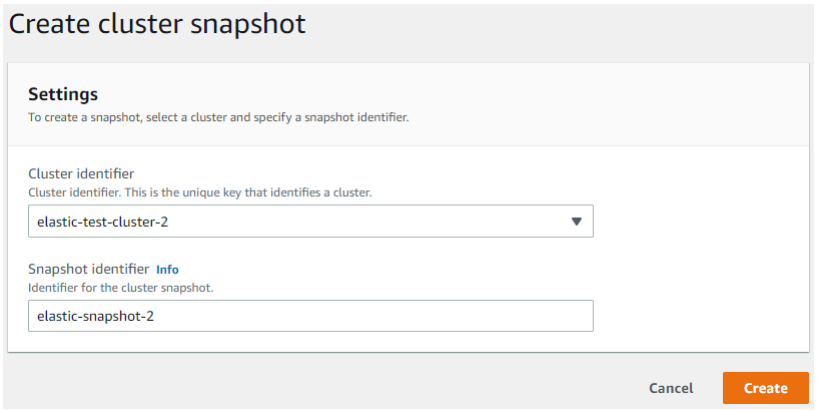
1. Melden Sie sich bei der Amazon DocumentDB-Konsole an AWS-Managementkonsole und öffnen Sie sie.
2. Wählen Sie im Navigationsbereich die Option Snapshots.

   > **Tip**
   >
   > Wenn Sie den Navigationsbereich nicht auf der linken Seite Ihres Bildschirms sehen, wählen Sie das Menüsymbol in der oberen linken Ecke des Navigationsbereichs.

3. Wählen Sie auf der Seite Snapshots die Option Create (Erstellen) aus.
4. Wählen Sie auf der Seite „Cluster-Snapshot erstellen“ im Feld Cluster-ID Ihren Elastic Cluster aus der Dropdownliste aus.

   Geben Sie im Feld Snapshot-ID eine eindeutige Kennung für Ihren Elastic Cluster ein.

   Wählen Sie Erstellen aus.



> **Note**
>
> Sie können auch auf das Dialogfeld „Cluster-Snapshot erstellen“ zugreifen, indem Sie auf die Seite „Cluster“ gehen, das Kästchen neben Ihrem Cluster markieren und dann „Aktionen“ und dann „Snapshot erstellen“ auswählen.

Ihr Elastic Cluster-Snapshot wird jetzt bereitgestellt. Es kann bis zu ein paar Minuten dauern, bis dieser Vorgang abgeschlossen ist. Sie können Ihren Snapshot anzeigen und aus ihm wiederherstellen, wenn der Status wie `Available` in der Snapshot-Liste angezeigt wird.

Using the AWS CLI

Verwenden Sie den `create-cluster-snapshot` Vorgang mit den folgenden Parametern AWS CLI, um einen manuellen Elastic-Cluster-Snapshot mithilfe von zu erstellen:

- **`--snapshot-name`**—Erforderlich. Der Name des Cluster-Snapshots, den Sie erstellen möchten.
- **`--cluster-arn`**—Erforderlich. Die ARN-ID des Clusters, von dem Sie einen Snapshot erstellen möchten.

Ersetzen Sie im folgenden Beispiel jedes *`user input placeholder`* durch Ihre eigenen Informationen.

Für Linux, macOS oder Unix:

```
aws docdb-elastic create-cluster-snapshot \
    --snapshot-name sample-snapshot-1 \
```

```
    --cluster-arn arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-
cluster
```

Für Windows:

```
aws docdb-elastic create-cluster-snapshot ^
    --snapshot-name sample-snapshot-1 ^
    --cluster-arn arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-
cluster
```

Einen Elastic Cluster-Snapshot anzeigen

In diesem Abschnitt erklären wir, wie Sie Elastic Cluster-Snapshot-Informationen anzeigen, indem Sie entweder die AWS-Managementkonsole oder AWS CLI die folgenden Anweisungen verwenden.

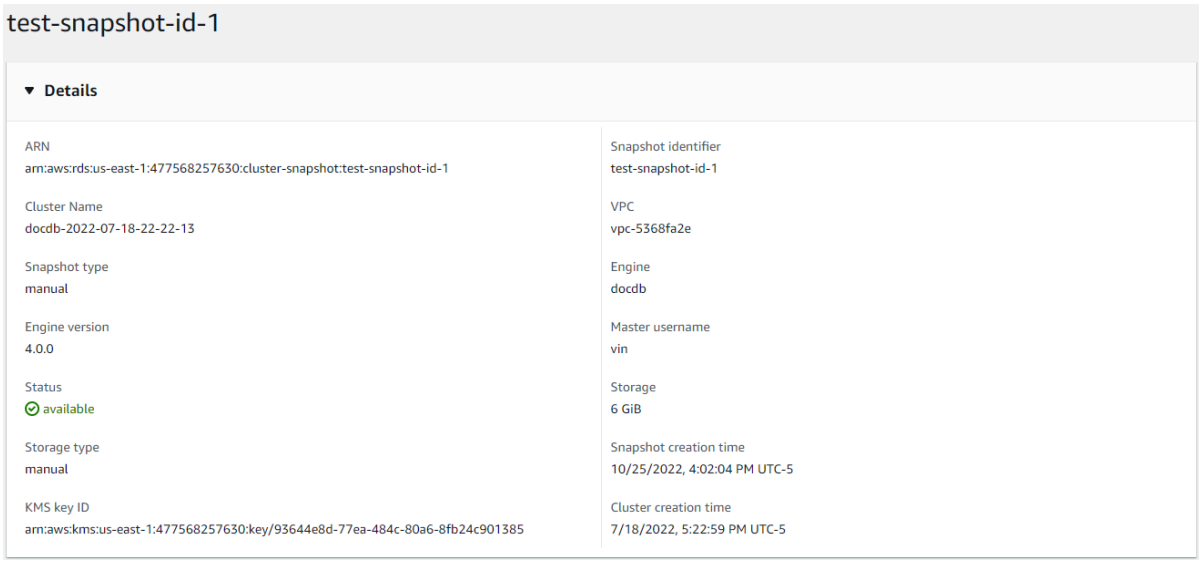
Using the AWS-Managementkonsole

Um Informationen zu einem bestimmten Elastic Cluster-Snapshot anzuzeigen, verwenden Sie AWS-Managementkonsole:

1. Melden Sie sich bei der Amazon DocumentDB-Konsole an AWS-Managementkonsole und öffnen Sie sie.
2. Wählen Sie im Navigationsbereich die Option Snapshots.

   > **Tip**
   >
   > Wenn Sie den Navigationsbereich nicht auf der linken Seite Ihres Bildschirms sehen, wählen Sie das Menüsymbol in der oberen linken Ecke des Navigationsbereichs.

3. Wählen Sie auf der Seite Snapshots Ihren Snapshot aus der Liste aus, indem Sie auf den Namen in der Spalte Snapshot-ID klicken.
4. Sehen Sie sich die Informationen zu Ihrem Snapshot unter Details an.



Using the AWS CLI

Um Informationen zu einem bestimmten Elastic Cluster-Snapshot mithilfe von anzuzeigen AWS CLI, verwenden Sie den `get-cluster-snapshot` Vorgang mit den folgenden Parametern:

- **`--snapshot-arn`**—Erforderlich. Die ARN-ID des Snapshots, für den Sie Informationen benötigen.

Ersetzen Sie im folgenden Beispiel jede *`user input placeholder`* durch Ihre eigenen Informationen.

Für Linux, macOS oder Unix:

```
aws docdb-elastic get-cluster-snapshot \
    --snapshot-arn sampleResourceName
```

Für Windows:

```
aws docdb-elastic get-cluster-snapshot ^
    --snapshot-arn sampleResourceName
```

Um Informationen zu einem bestimmten Elastic Cluster-Snapshot mithilfe von anzuzeigen AWS CLI, verwenden Sie den `get-cluster-snapshot` Vorgang mit den folgenden Parametern:

- **`--snapshot-arn`**—Erforderlich. Die ARN-ID des Snapshots, für den Sie Informationen benötigen.

Ersetzen Sie im folgenden Beispiel jede *`user input placeholder`* durch Ihre eigenen Informationen.

Für Linux, macOS oder Unix:

```
aws docdb-elastic get-cluster-snapshot \
    --snapshot-arn sampleResourceName
```

Für Windows:

```
aws docdb-elastic get-cluster-snapshot ^
    --snapshot-arn sampleResourceName
```

Um Informationen zu allen Elastic Cluster-Snapshots mithilfe von anzuzeigen AWS CLI, verwenden Sie den `list-cluster-snapshots` Vorgang mit den folgenden Parametern:

- **`--snapshot-type`**— Fakultativ. Der Typ der Cluster-Snapshots, die zurückgegeben werden sollen. Sie können einen der folgenden Werte angeben:
  - `automated`— Gibt alle Cluster-Snapshots zurück, die Amazon DocumentDB automatisch für Ihr Konto erstellt hat. AWS
  - `manual`- Gibt alle Cluster-Snapshots zurück, die Sie manuell für Ihr Konto erstellt haben. AWS
  - `shared`- Gibt alle manuellen Cluster-Snapshots zurück, die für Ihr AWS Konto freigegeben wurden.
  - `public`- Gibt alle Cluster-Snapshots zurück, die als öffentlich markiert wurden.
- **`--next-token`**— Fakultativ. Ein optionales Paginierungstoken, das von einer vorherigen Anforderung bereitgestellt wird. Wenn dieser Parameter angegeben ist, enthält die Antwort nur Datensätze, die über dieses Token hinausgehen, bis zu dem von `max-results` angegebenen Wert.
- **`--max-results`**— Fakultativ. Die maximale Anzahl von Ergebnissen, die in die Antwort aufgenommen werden sollen. Wenn mehr Ergebnisse als der angegebene `max-results` Wert

vorhanden sind, ist ein Paginierungstoken (`next-token`) in der Antwort enthalten, sodass die verbleibenden Ergebnisse abgerufen werden können.

- Standard: 100
- Mindestens 20, maximal 100

Ersetzen Sie im folgenden Beispiel jedes *`user input placeholder`* durch Ihre eigenen Informationen.

Für Linux, macOS oder Unix:

```
aws docdb-elastic list-cluster-snapshots \
    --snapshot-type  value \
    --next-token  value \
    --max-results 50
```

Für Windows:

```
aws docdb-elastic list-cluster-snapshots ^
    --snapshot-type  value ^
    --next-token  value ^
    --max-results 50
```

Wiederherstellen eines elastischen Clusters aus einem Snapshot

In diesem Abschnitt erklären wir, wie Sie einen elastischen Cluster aus einem Snapshot wiederherstellen, indem Sie entweder die AWS-Managementkonsole oder AWS CLI die folgenden Anweisungen verwenden.

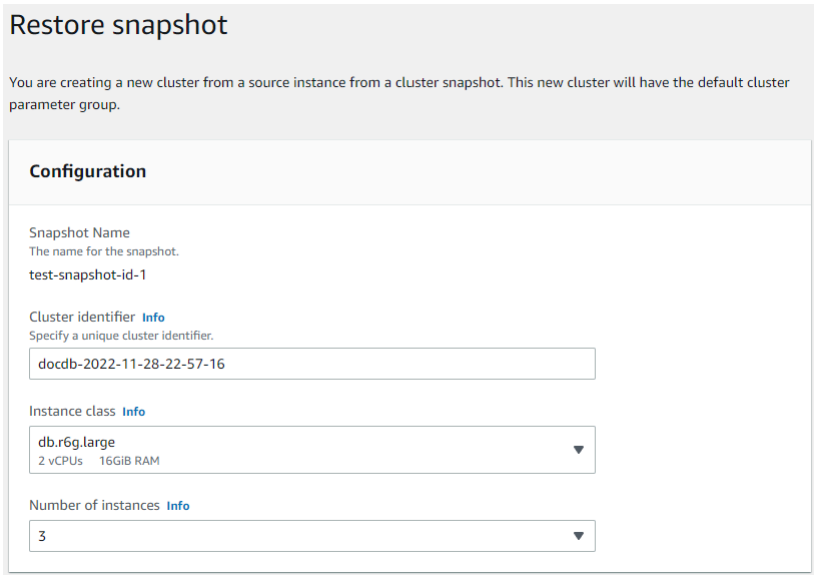
Using the AWS-Managementkonsole

Um einen elastischen Cluster aus einem Snapshot wiederherzustellen, verwenden Sie AWS-Managementkonsole:

1. Melden Sie sich bei der Amazon DocumentDB-Konsole an AWS-Managementkonsole und öffnen Sie sie.
2. Wählen Sie im Navigationsbereich die Option Snapshots.

> **Tip**
>
> Wenn Sie den Navigationsbereich nicht auf der linken Seite Ihres Bildschirms sehen, wählen Sie das Menüsymbol in der oberen linken Ecke des Navigationsbereichs.

3. Wählen Sie in der Spalte Snapshot-ID die Schaltfläche links neben dem Snapshot, den Sie zum Wiederherstellen eines Clusters verwenden möchten.
4. Wählen Sie Aktionen und dann Wiederherstellen aus.



5. Geben Sie auf der Seite „Snapshot wiederherstellen“ einen Namen für den neuen Cluster in das Feld Cluster-ID ein.

> **Note**
>
> Für jede manuelle Snapshot-Wiederherstellung müssen Sie einen neuen Cluster erstellen.

6. Wählen Sie im Feld Virtual Private Cloud (VPC) eine VPC aus der Dropdownliste aus.
7. Für Subnetze und VPC-Sicherheitsgruppen können Sie die Standardeinstellungen verwenden oder drei Subnetze Ihrer Wahl und bis zu drei VPC-Sicherheitsgruppen (mindestens eine) auswählen.
8. Wenn Sie mit der Cluster-Konfiguration zufrieden sind, wählen Sie Restore cluster (Cluster wiederherstellen) aus und warten Sie, bis Ihr Cluster wiederhergestellt ist.

Using the AWS CLI

Um einen elastischen Cluster mithilfe von aus einem Snapshot wiederherzustellen AWS CLI, verwenden Sie den Vorgang mit den folgenden Parametern: `restore-cluster-from-snapshot`

- **`--cluster-name`**—Erforderlich. Der aktuelle Name des elastischen Clusters, wie er bei der Erstellung eingegeben oder zuletzt geändert wurde.
- **`--snapshot-arn`**—Erforderlich. Die ARN-ID des Snapshots, der zur Wiederherstellung des Clusters verwendet wird.
- **`--vpc-security-group-ids`**— Fakultativ. Eine oder mehrere Amazon EC2- und Amazon Virtual Private Cloud (VPC) -Sicherheitsgruppen, die dem Cluster zugeordnet werden sollen.
- **`--kms-key-id`**— Fakultativ. Konfigurieren Sie die KMS-Schlüssel-ID für einen verschlüsselten Cluster.

  Die KMS-Schlüssel-ID ist der Amazon-Ressourcenname (ARN) für den AWS KMS Verschlüsselungsschlüssel. Wenn Sie einen Cluster mit demselben Amazon Web Services-Konto erstellen, dem der KMS-Verschlüsselungsschlüssel gehört, der zum Verschlüsseln des neuen Clusters verwendet wird, können Sie den KMS-Schlüsselalias anstelle des ARN für den KMS-Verschlüsselungsschlüssel verwenden.

  Wenn in diesem Feld kein Verschlüsselungsschlüssel angegeben ist KmsKeyId und der `StorageEncrypted` Parameter wahr ist, verwendet Amazon DocumentDB Ihren Standard-Verschlüsselungsschlüssel.
- **`--subnet-ids`**— Fakultativ. Netzwerk-Subnetz-IDs.

Ersetzen Sie im folgenden Beispiel jede *`user input placeholder`* durch Ihre eigenen Informationen.

Für Linux, macOS oder Unix:

```
aws docdb-elastic restore-cluster-from-snapshot \
     --cluster-name elastic-sample-cluster \
     --snapshot-arn sampleResourceName \
     --vpc-security-group-ids value ec-65f40350 \
     --kms-key-id arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2 \
     --subnet-ids subnet-9253c6a3, subnet-9f1b5af9
```

Für Windows:

```
aws docdb-elastic restore-cluster-from-snapshot ^
     --cluster-name elastic-sample-cluster ^
     --snapshot-arn sampleResourceName ^
     --vpc-security-group-ids value ec-65f40350 ^
     --kms-key-id arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2 ^
     --subnet-ids subnet-9253c6a3, subnet-9f1b5af9
```

Einen Elastic Cluster-Snapshot kopieren

In Amazon DocumentDB können Sie manuelle und automatische Elastic Cluster-Snapshots innerhalb derselben Region und innerhalb desselben Kontos kopieren. In diesem Abschnitt erklären wir, wie Sie einen Elastic-Cluster-Snapshot kopieren, indem Sie entweder das AWS-Managementkonsole oder verwenden. AWS CLI

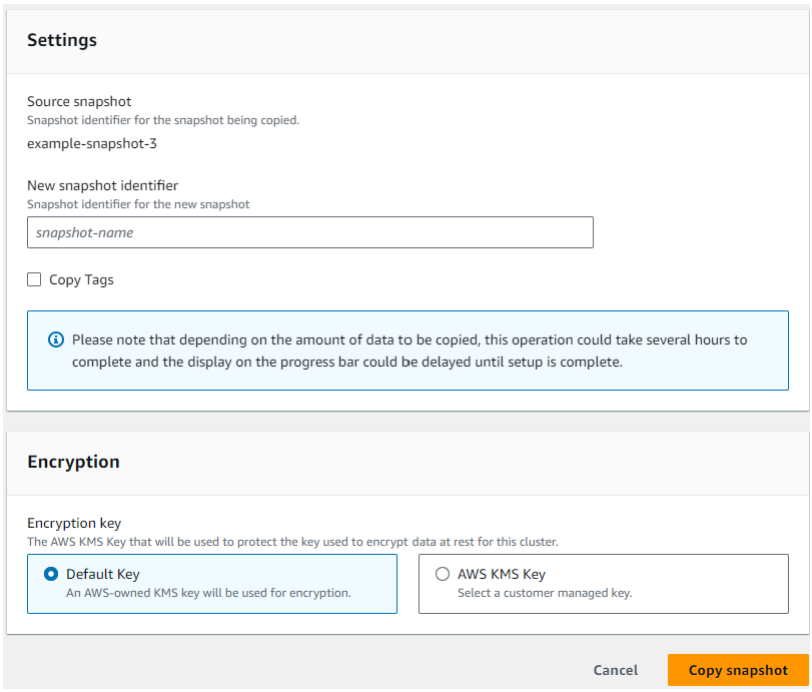
Using the AWS-Managementkonsole

Um einen Elastic Cluster-Snapshot zu kopieren, verwenden Sie AWS-Managementkonsole:

1. Melden Sie sich bei der Amazon DocumentDB-Konsole an AWS-Managementkonsole und öffnen Sie sie.
2. Wählen Sie im Navigationsbereich die Option Snapshots.

   > **Tip**
   >
   > Wenn Sie den Navigationsbereich nicht auf der linken Seite Ihres Bildschirms sehen, wählen Sie das Menüsymbol in der oberen linken Ecke des Navigationsbereichs.

3. Wählen Sie in der Spalte Snapshot-ID die Schaltfläche links neben dem Snapshot, den Sie kopieren möchten.
4. Wählen Sie Aktionen und dann Kopieren.



5. Geben Sie unter Neue Snapshot-ID den Namen des neuen Snapshots ein.
6. Markieren Sie bei Copy Tags das Kästchen, wenn Sie alle Tags aus dem Elastic Cluster-Quell-Snapshot in den Elastic Cluster-Ziel-Snapshot kopieren möchten.
7. Wählen Sie für Verschlüsselung entweder einen AWS Standard-KMS-Schlüssel oder einen KMS-Schlüssel Ihrer Wahl. Mit der zweiten Option können Sie einen vorhandenen KMS-Schlüssel auswählen, den Sie bereits erstellt haben, oder Sie können einen neuen erstellen.
8. Wählen Sie Snapshot kopieren, wenn Sie fertig sind.

Using the AWS CLI

Um einen Elastic Cluster-Snapshot mithilfe von zu kopieren AWS CLI, verwenden Sie den `copy-cluster-snapshot` Vorgang mit den folgenden Parametern:

- **`--source-db-cluster-snapshot-identifier`**—Erforderlich. Die ID des vorhandenen Elastic Cluster-Snapshots, der kopiert wird. Der Elastic Cluster-Snapshot muss vorhanden sein und sich im Status „Verfügbar“ befinden. Bei diesem Parameter wird nicht zwischen Groß- und Kleinschreibung unterschieden.
- **`--target-db-cluster-snapshot-identifier`**—Erforderlich. Die ID des neuen elastischen Cluster-Snapshots, der aus dem vorhandenen Cluster-Snapshot erstellt werden soll. Bei diesem Parameter wird nicht zwischen Groß- und Kleinschreibung unterschieden.

Einschränkungen für den Namen des Ziel-Snapshots:

- Kann nicht der Namen eines vorhandenen Snapshots sein.
- Die Länge beträgt [1—63] Buchstaben, Zahlen oder Bindestriche.
- Muss mit einem Buchstaben beginnen.
- Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.

Ersetzen Sie im folgenden Beispiel jedes durch Ihre *user input placeholder* eigenen Informationen.

Für Linux, macOS oder Unix:

```
aws docdb-elastic copy-cluster-snapshot \
     --source-cluster-snapshot-arn <sample ARN> \
     --target-cluster-snapshot-name my-target-copied-snapshot
```

Für Windows:

```
aws docdb-elastic copy-cluster-snapshot ^
     --source-cluster-snapshot-arn <sample ARN> ^
     --target-cluster-snapshot-name my-target-copied-snapshot
```

Löschen eines elastischen Cluster-Snapshots

In diesem Abschnitt erklären wir, wie Sie einen Elastic-Cluster-Snapshot löschen, indem Sie entweder das AWS-Managementkonsole oder verwenden AWS CLI.

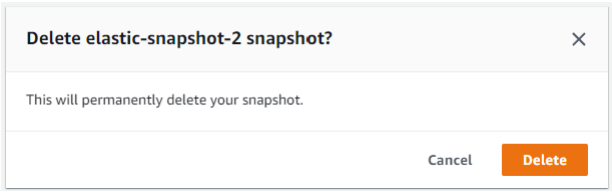
Using the AWS-Managementkonsole

Um einen elastischen Cluster aus einem Snapshot wiederherzustellen, verwenden Sie AWS-Managementkonsole:

1. Melden Sie sich bei der Amazon DocumentDB-Konsole an AWS-Managementkonsole und öffnen Sie sie.
2. Wählen Sie im Navigationsbereich die Option Snapshots.

> **Tip**
>
> Wenn Sie den Navigationsbereich nicht auf der linken Seite Ihres Bildschirms sehen, wählen Sie das Menüsymbol in der oberen linken Ecke des Navigationsbereichs.

3. Wählen Sie in der Spalte Snapshot-ID die Schaltfläche links neben dem Snapshot, den Sie zum Wiederherstellen eines Clusters verwenden möchten.
4. Wählen Sie Aktionen und dann Löschen aus.



5. Wählen Sie im Dialogfeld „Snapshot-Name“ -Snapshot löschen die Option Löschen aus.

Using the AWS CLI

Um einen Elastic Cluster-Snapshot mithilfe von zu löschen AWS CLI, verwenden Sie den `delete-cluster-snapshot` Vorgang mit den folgenden Parametern:

- **`--snapshot-arn`**—Erforderlich. Die ARN-ID des Snapshots, der zur Wiederherstellung des Clusters verwendet wird.

Ersetzen Sie im folgenden Beispiel jedes *`user input placeholder`* durch Ihre eigenen Informationen.

Für Linux, macOS oder Unix:

```
aws docdb-elastic delete-cluster-snapshot \
     --snapshot-arn sampleResourceName
```

Für Windows:

```
aws docdb-elastic delete-cluster-snapshot ^
     --snapshot-arn sampleResourceName
```

Verwaltung eines automatischen Backups eines Elastic Cluster-Snapshots

Amazon DocumentDB erstellt täglich Snapshots Ihrer Elastic Cluster. Sie können das bevorzugte Backup-Fenster und den Aufbewahrungszeitraum für Backups in einer neuen oder vorhandenen Elastic Cluster-Snapshot-Konfiguration angeben. In diesem Abschnitt erklären wir, wie Sie automatische Backup-Parameter in einem Elastic-Cluster-Snapshot festlegen, indem Sie entweder das AWS-Managementkonsole oder verwenden AWS CLI.

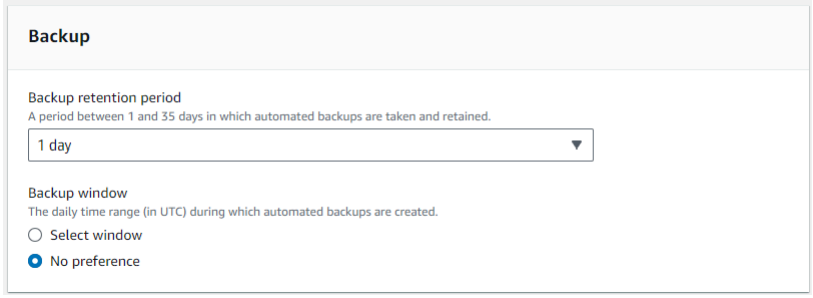
Using the AWS-Managementkonsole

Um ein automatisches Backup für einen neuen Elastic Cluster-Snapshot einzurichten, verwenden Sie AWS-Managementkonsole:

1. Melden Sie sich bei der Amazon DocumentDB-Konsole an AWS-Managementkonsole und öffnen Sie sie.
2. Klicken Sie im Navigationsbereich auf Cluster.

> **Tip**
>
> Wenn Sie den Navigationsbereich nicht auf der linken Seite Ihres Bildschirms sehen, wählen Sie das Menüsymbol in der oberen linken Ecke des Navigationsbereichs.

3. Wählen Sie in der Spalte Cluster-ID die Schaltfläche links neben dem Cluster, für den Sie die Backup-Einstellungen ändern möchten.
4. Wählen Sie Aktionen und dann Ändern aus.
5. Bearbeiten Sie im Abschnitt Backup die Felder entsprechend Ihren Backup-Anforderungen.



   a. Aufbewahrungszeitraum für Backups — Wählen Sie in der Liste die Anzahl der Tage aus, für die automatische Backups dieses Clusters aufbewahrt werden sollen, bevor sie gelöscht werden.

b. Backup-Fenster — Legen Sie die tägliche Uhrzeit und Dauer fest, während der Amazon DocumentDB Backups dieses Clusters erstellen soll.

   i. Wählen Sie Fenster auswählen, wenn Sie den Zeitpunkt und die Dauer der Erstellung von Backups konfigurieren möchten.

      Startzeit — Wählen Sie in der ersten Liste die Startzeit (Stunde) (UTC) für den Start Ihrer automatischen Backups aus. Wählen Sie in der zweiten Liste die Minute für den Beginn der automatischen Backups aus.

      Dauer — Wählen Sie in der Liste die Anzahl der Stunden aus, die der Erstellung automatischer Backups zugewiesen werden sollen.

   ii. Wählen Sie Keine Präferenz, wenn Amazon DocumentDB den Zeitpunkt und die Dauer der Erstellung von Backups auswählen soll.

6. Wählen Sie Cluster ändern, wenn Sie fertig sind.

Using the AWS CLI

Um ein automatisches Backup für einen neuen Elastic Cluster-Snapshot mithilfe von einzurichten AWS CLI, verwenden Sie den `create-cluster-snapshot` Vorgang mit den folgenden Parametern:

- **`--preferred-backup-window`**— Fakultativ. Der bevorzugte tägliche Zeitraum, in dem automatische Backups erstellt werden. Die Standardeinstellung ist ein 30-minütiges Fenster, das nach dem Zufallsprinzip aus einem Zeitblock von jeweils AWS-Region 8 Stunden ausgewählt wird.

  Einschränkungen:

  - Muss im Format `hh24:mi-hh24:mi` angegeben werden.
  - Muss in Universal Coordinated Time (UTC) angegeben werden.
  - Darf nicht mit dem bevorzugten Wartungsfenster in Konflikt treten.
  - Muss mindestens 30 Minuten betragen.
- **`--backup-retention-period`**— Fakultativ. Die Anzahl von Tagen, über die automatische Backups aufbewahrt werden. Der Standardwert ist 1.

  Einschränkungen:

  - Es muss ein Mindestwert von 1 angegeben werden.

- Der Bereich liegt zwischen 1 und 35.

> **Note**
>
> Automatische Backups werden nur erstellt, wenn sich der Cluster in einem „aktiven“ Zustand befindet.

> **Note**
>
> Sie können auch die `backup-retention-period` Parameter `preferred-backup-window` und eines vorhandenen Elastic Clusters mithilfe des `aws docdb-elastic update-cluster` Befehls ändern.

Ersetzen Sie im folgenden Beispiel jedes *`user input placeholder`* durch Ihre eigenen Informationen.

Das folgende `create-cluster` Beispiel erstellt den Amazon DocumentDB Elastic Cluster *`sample-cluster`* mit einer Aufbewahrungsfrist für automatische Backups von *`7`* Tagen und einem bevorzugten Backup-Fenster von*`18:00-18:30 UTC`*.

Für Linux, macOS oder Unix:

```
aws docdb-elastic create-cluster \
     --cluster-name sample-cluster \
     --shard-capacity 2 \
     --shard-count 2 \
     --admin-user-name SampleAdmin \
     --auth-type PLAIN_TEXT \
     --admin-user-password SamplePass123! \
     --preferred-backup-window 18:00-18:30 \
     --backup-retention-period 7
```

Für Windows:

```
aws docdb-elastic create-cluster ^
     --cluster-name sample-cluster ^
     --shard-capacity 2 ^
```

```
    --shard-count 2 ^
    --admin-user-name SampleAdmin ^
    --auth-type PLAIN_TEXT ^
    --admin-user-password SamplePass123! ^
    --preferred-backup-window 18:00-18:30 ^
    --backup-retention-period 7
```

Stoppen und Starten eines elastischen Amazon DocumentDB-Clusters

Das Stoppen und Starten von Amazon DocumentDB Elastic Clusters kann Ihnen helfen, die Kosten für Entwicklungs- und Testumgebungen zu kontrollieren. Anstatt bei jeder Verwendung von Amazon DocumentDB elastische Cluster zu erstellen und zu löschen, können Sie Ihren Cluster vorübergehend anhalten, wenn er nicht benötigt wird. Sie können ihn dann erneut starten, wenn Sie Ihre Tests fortsetzen.

Themen

- Überblick über das Stoppen und Starten eines elastischen Clusters
- Operationen, die Sie auf einem gestoppten Elastic Cluster ausführen können

Überblick über das Stoppen und Starten eines elastischen Clusters

In Zeiten, in denen Sie keinen elastischen Amazon DocumentDB-Cluster benötigen, können Sie den Cluster beenden. Bei Bedarf können Sie den Cluster jederzeit erneut starten. Das Starten und Stoppen vereinfacht die Einrichtung und Demontage für elastische Cluster, die für Entwicklungs-, Test- oder ähnliche Aktivitäten verwendet werden, für die keine kontinuierliche Verfügbarkeit erforderlich ist. Sie können einen elastischen Cluster mithilfe von AWS-Managementkonsole oder AWS CLI mit einer einzigen Aktion beenden und starten.

Solange Ihr Elastic Cluster gestoppt ist, bleibt das Cluster-Speicher-Volume unverändert. Sie zahlen nur für Speicherung, manuelle Snapshots und automatischen Sicherungsspeicher innerhalb des angegebenen Aufbewahrungsfensters. Amazon DocumentDB startet Ihren Elastic Cluster automatisch nach sieben Tagen, sodass er nicht aufgrund der erforderlichen Wartungsupdates ausfällt. Wenn Ihr Cluster nach sieben Tagen startet, wird Ihnen die Nutzung des Elastic Clusters wieder in Rechnung gestellt. Solange Ihr Cluster gestoppt ist, können Sie Ihr Speichervolume nicht abfragen, da für die Abfrage der Status „Verfügbar“ erforderlich ist.

Wenn ein elastischer Amazon DocumentDB-Cluster gestoppt wird, kann der Cluster in keiner Weise geändert werden. Dazu gehört auch das Löschen des Clusters.

Using the AWS-Managementkonsole

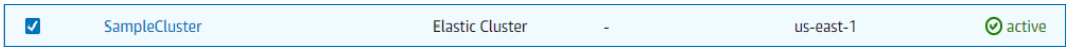
Das folgende Verfahren zeigt Ihnen, wie Sie einen elastischen Cluster im verfügbaren Zustand stoppen oder einen gestoppten elastischen Cluster starten.

Um einen elastischen Amazon DocumentDB-Cluster zu stoppen oder zu starten

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon DocumentDB-Konsole unter. https://console.aws.amazon.com/docdb
2. Klicken Sie im Navigationsbereich auf Cluster.

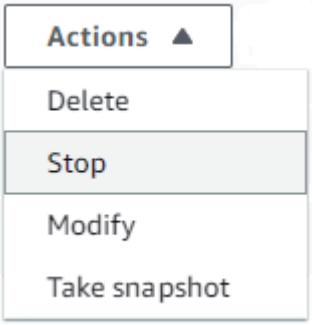
   > **Tip**
   >
   > Wenn der Navigationsbereich auf der linken Seite des Bildschirms nicht angezeigt wird, wählen Sie links oben auf der Seite das Menüsymbol (≡) aus.

3. Wählen Sie in der Liste der Cluster die Schaltfläche links neben dem Namen des Clusters aus, den Sie stoppen oder starten möchten.

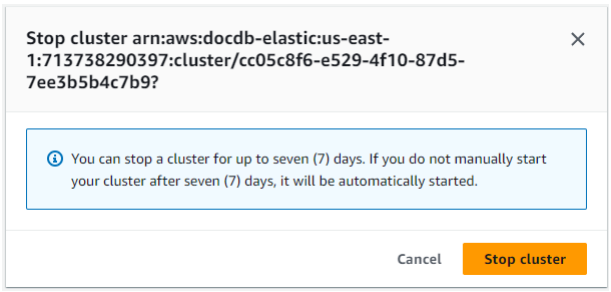
   

4. Wählen Sie Aktionen aus und dann die Aktion, die Sie auf dem Cluster ausführen möchten.
   - Wenn Sie den Cluster stoppen möchten und der Cluster verfügbar ist:

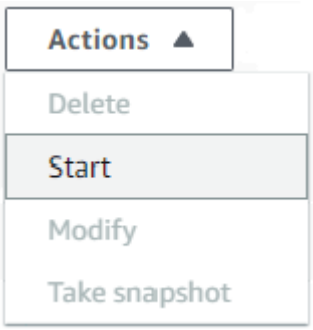
     a. Wählen Sie Beenden aus.

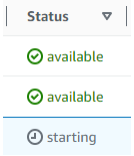
     b. Bestätigen Sie im Bestätigungsdialogfeld, dass Sie den Elastic Cluster stoppen möchten, indem Sie Cluster stoppen wählen, oder wählen Sie Abbrechen, um den Cluster am Laufen zu halten.



- Wenn Sie einen gestoppten Cluster starten möchten, wählen Sie Starten aus.



5. Überwachen Sie den Status des elastischen Clusters. Wenn Sie den Cluster gestartet haben, können Sie den Cluster wieder verwenden, sobald der Cluster verfügbar ist. Weitere Informationen finden Sie unter Den Status eines Clusters ermitteln.



Using the AWS CLI

Die folgenden Codebeispiele zeigen Ihnen, wie Sie einen elastischen Cluster im aktiven oder verfügbaren Zustand stoppen oder einen gestoppten elastischen Cluster starten.

Verwenden Sie den `stop-cluster` Vorgang AWS CLI, um einen elastischen Cluster mithilfe von zu stoppen. Verwenden Sie die `start-cluster`-Operation, um einen gestoppten Cluster zu starten. Beide Operationen verwenden den `--cluster-arn`-Parameter.

Parameter:

- **`--cluster-arn`**—Erforderlich. Die ARN-ID des elastischen Clusters, den Sie beenden oder starten möchten.

Example— Um einen elastischen Cluster zu stoppen, verwenden Sie den AWS CLI

Ersetzen Sie im folgenden Beispiel jedes *user input placeholder* durch Ihre eigenen Informationen.

Der folgende Code stoppt den elastischen Cluster mit einem ARN von`arn:aws:docdb-elastic:us-east-1:477568257630:cluster/b9f1d489-6c3e-4764-bb42-da62ceb7bda2`.

> **Note**
>
> Der elastische Cluster muss sich im Status „Aktiv“ oder „Verfügbar“ befinden.

Für Linux, macOS oder Unix:

```
aws docdb-elastic stop-cluster \
   --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2
```

Für Windows:

```
aws docdb-elastic stop-cluster ^
   --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2
```

Example— Um einen elastischen Cluster zu starten, verwenden Sie den AWS CLI

Ersetzen Sie im folgenden Beispiel jedes *user input placeholder* durch Ihre eigenen Informationen.

Der folgende Code startet den elastischen Cluster mit einem ARN von`arn:aws:docdb-elastic:us-east-1:477568257630:cluster/b9f1d489-6c3e-4764-bb42-da62ceb7bda2`.

> **Note**
>
> Der elastische Cluster muss derzeit gestoppt werden.

Für Linux, macOS oder Unix:

```
aws docdb-elastic start-cluster \
   --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2
```

Für Windows:

```
aws docdb-elastic start-cluster ^
   --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2
```

Operationen, die Sie auf einem gestoppten Elastic Cluster ausführen können

Sie können die Konfiguration eines elastischen Amazon DocumentDB-Clusters nicht ändern, solange der Cluster gestoppt ist. Vor solchen administrativen Aktionen müssen Sie den Cluster starten.

Amazon DocumentDB wendet alle geplanten Wartungsarbeiten auf Ihren gestoppten Elastic Cluster erst an, nachdem er erneut gestartet wurde. Nach sieben Tagen startet Amazon DocumentDB automatisch einen gestoppten Elastic Cluster, sodass dieser in seinem Wartungsstatus nicht zu weit in Verzug gerät. Wenn der Elastic Cluster neu gestartet wird, werden Ihnen die Shards im Cluster wieder in Rechnung gestellt.

Solange ein elastischer Cluster gestoppt wird, führt Amazon DocumentDB keine automatischen Backups durch und verlängert auch nicht die Aufbewahrungsfrist für Backups.

Wartung der elastischen Amazon DocumentDB-Cluster

Topics

- Ausstehende Wartungsmaßnahmen für elastische Cluster anzeigen
- Aktualisierungen der Elastic Cluster Engine
- Aktualisierungen des Elastic Cluster-Betriebssystems

In regelmäßigen Abständen führt Amazon DocumentDB Wartungsarbeiten an den elastischen Cluster-Ressourcen von Amazon DocumentDB durch. Die Wartung beinhaltet meistens Aktualisierungen der Datenbank-Engine (Elastic Cluster Maintenance) oder des dem Elastic Cluster zugrunde liegenden Betriebssystems (Betriebssystem-Updates). Datenbank-Engine-Updates

sind erforderliche Patches und beinhalten Sicherheitsupdates, Bugfixes und Verbesserungen der Datenbank-Engine. Die meisten Betriebssystem-Patches sind zwar optional, aber wenn Sie sie eine Zeit lang nicht anwenden, ist der Patch möglicherweise erforderlich und wird automatisch angewendet, um Ihre Sicherheitslage aufrechtzuerhalten. Wenden Sie Betriebssystem-Updates auf Ihre Amazon DocumentDB Elastic Cluster an, sobald sie verfügbar sind.

Für Patches für Datenbank-Engines müssen Sie Ihre Amazon DocumentDB Elastic Clusters für kurze Zeit offline schalten. Sobald diese Patches verfügbar sind, werden sie automatisch während eines bevorstehenden geplanten Wartungsfensters Ihres Amazon DocumentDB Elastic Clusters installiert.

Elastische Cluster haben jeweils ihre eigenen Wartungsfenster. Elastische Cluster-Änderungen, die Sie nicht sofort anwenden möchten, werden während des Wartungsfensters angewendet. Wenn Sie einen Elastic Cluster erstellen, weist Amazon DocumentDB Ihrem Elastic Cluster standardmäßig ein Wartungsfenster zu. Sie können das Wartungsfenster wählen, wenn Sie einen elastischen Cluster erstellen. Sie können die Wartungsfenster außerdem jederzeit an Ihre Geschäftspläne oder -praktiken anpassen. Es wird generell empfohlen, Wartungsfenster auszuwählen, die die Auswirkungen der Wartung auf Ihre Anwendung minimieren (z. B. an Abenden oder Wochenenden).

Ausstehende Wartungsmaßnahmen für elastische Cluster anzeigen

Sie können überprüfen, ob ein Wartungsupdate für Ihren Elastic Cluster verfügbar ist, indem Sie das verwenden AWS CLI.

Wenn ein Update verfügbar ist, können Sie eine der folgenden Aktionen durchführen:

- Verschieben Sie eine Wartungsmaßnahme, die derzeit für das nächste Wartungsfenster geplant ist (nur für Betriebssystem-Patches).
- Wenden Sie die Wartungsaktionen sofort an.
- Einplanen der Wartungsaktivitäten für das nächste Wartungsfenster.
- Planen Sie die Wartungsmaßnahmen so, dass sie während des von Ihnen ausgewählten Anwendungsfensters beginnen.

Das Wartungsfenster legt fest, wann die ausstehenden Operationen gestartet werden, gibt aber kein Abschlussdatum für diese Operationen vor; die Ausführungsdauer ist weder bekannt noch a priori beschränkt.

Verwenden Sie den folgenden AWS CLI Vorgang, um zu ermitteln, welche Wartungsmaßnahmen noch ausstehen. Listet alle ausstehenden Wartungsmaßnahmen auf:

```
aws docdb-elastic list-pending-maintenance-actions
```

Die Ausgabe dieses Vorgangs sieht ungefähr wie folgt aus (JSON-Format):

```
{
'ResourcePendingMaintenanceActions': [
    {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'ENGINE_UPDATE',
                'AutoAppliedAfterDate': 'string',
                'ForcedApplyDate': 'string',
                'OptInStatus': 'string',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            },
        ]
    },
],
'NextToken': 'string'
}
```

Ruft ausstehende Wartungsmaßnahmen (falls vorhanden) für eine bestimmte Datei ab`resourceArn`:

```
aws docdb-elastic get-pending-maintenance-action --resource-arn string-arn
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    'ResourcePendingMaintenanceAction': {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'ENGINE_UPDATE',
                'AutoAppliedAfterDate': 'string',
                'ForcedApplyDate': 'string',
                'OptInStatus': 'string',
                'CurrentApplyDate': 'string',
                'Description': 'string'
```

```
            }
        ]
    }
}
```

Parameter:

- `ResourceArn`— Der Amazon DocumentDB Amazon-Ressourcenname (ARN) der Ressource, für die sich die ausstehende Wartungsmaßnahme bezieht.
- `Action`— Die ausstehende Wartungsmaßnahme, die auf die Ressource angewendet wird.

  Zulässige Werte:

  - `ENGINE_UPDATE`
  - `ENGINE_UPGRADE`
  - `SECURITY_UPDATE`
  - `OS_UPDATE`
  - `MASTER_USER_PASSWORD_UPDATE`
- `AutoAppliedAfterDate`— Erstes Wartungsfenster nach diesem Datum. `NEXT_MAINTENANCE` `OPT_IN`wird in diesem Fall ignoriert.
- `ForcedApplyDate`— Wird unabhängig vom Wartungsfenster angewendet. `IMMEDIATE` `OPT_IN`wird in diesem Fall ignoriert.
- `OptInStatus`— Ein Wert, der den Typ der Opt-in-Anfrage angibt oder eine Opt-In-Anfrage rückgängig macht. Eine Opt-in-Anfrage vom Typ `IMMEDIATE` kann nicht rückgängig gemacht werden.

  Zulässige Werte:

  - `IMMEDIATE`— Wenden Sie die Wartungsmaßnahme sofort an.
  - `NEXT_MAINTENANCE`— Wenden Sie die Wartungsaktion im nächsten Wartungsfenster für die Ressource an.
  - `APPLY_ON`— Wendet die Wartungsaktion am angegebenen Anwendungsdatum an, unabhängig vom nächsten Wartungsfenster für die Ressource.
  - `UNDO_OPT_IN`— Storniert alle bestehenden Anfragen `NEXT_MAINTENANCE` oder `APPLY_ON` Opt-in-Anfragen.
- `CurrentApplyDate`— Wird angezeigt, wenn Opt-In-Type lautet. `APPLY_ON`
- `Description`— Eine Optionsbeschreibung für die Wartungsaktion.

Aktualisierungen der Elastic Cluster Engine

Mit Amazon DocumentDB können Sie wählen, wann Wartungsarbeiten durchgeführt werden sollen. Mithilfe der können Sie entscheiden, wann Amazon DocumentDB Aktualisierungen durchführt. AWS CLI

Wenden Sie ausstehende Wartungsmaßnahmen an:

```
aws docdb-elastic apply-pending-maintenance-action
--resource-arn string-arn
--apply-action string-enum
--opt-in-type string-enum
[--apply-on string-date-range]
```

Parameter:

- **`--resource-arn`**— Der Amazon DocumentDB Amazon-Ressourcenname (ARN) der Ressource, für die sich die ausstehenden Wartungsmaßnahmen beziehen.
- **`--apply-action`**— Die ausstehende Wartungsmaßnahme, die für diese Ressource gelten soll.

  Zulässige Werte:

  - `ENGINE_UPDATE`
  - `ENGINE_UPGRADE`
  - `SECURITY_UPDATE`
  - `OS_UPDATE`
  - `MASTER_USER_PASSWORD_UPDATE`
- **`--opt-in-type`**— Ein Wert, der den Typ der Opt-in-Anfrage angibt oder eine Opt-In-Anfrage rückgängig macht. Eine Opt-in-Anfrage vom Typ `IMMEDIATE` kann nicht rückgängig gemacht werden.

  Zulässige Werte:

  - `IMMEDIATE`— Wenden Sie die Wartungsmaßnahme sofort an.
  - `NEXT_MAINTENANCE`— Wenden Sie die Wartungsaktion im nächsten Wartungsfenster für die Ressource an.
  - `APPLY_ON`— Wendet die Wartungsaktion am angegebenen Anwendungsdatum an, unabhängig vom nächsten Wartungsfenster für die Ressource.

- UNDO_OPT_IN— Storniert alle bestehenden Anfragen NEXT_MAINTENANCE oder APPLY_ON Opt-in-Anfragen.
- **[--apply-on]**— Erforderlich, wenn Opt-In-Type ist. APPLY_ON Format: yyyy/MM/dd HH:mm-yyyy/MM/dd HH:mm (Diese Option verwendet die UTC-Zeit. Als Startzeit kann ein beliebiger Zeitpunkt in der Zukunft zwischen mindestens 30 Minuten und maximal 14 Tagen oder das force/apply Datum einer ausstehenden Aktion angegeben werden, je nachdem, welcher Zeitpunkt früher liegt. Das Zeitfenster von Anfang bis Ende kann mindestens 30 Minuten bis maximal 8 Stunden lang sein.)

Die Ausgabe dieses Vorgangs sieht ungefähr wie folgt aus (JSON-Format):

```
{
 'ResourcePendingMaintenanceAction': {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'SECURITY_UPDATE',
                'AutoAppliedAfterDate': 'string',
                'ForcedApplyDate': 'string',
                'OptInStatus': 'IMMEDIATE',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            },
        ]
 }
}
```

Parameter:

- ResourceArn— Der Amazon DocumentDB Amazon-Ressourcenname (ARN) der Ressource, für die die ausstehende Wartungsmaßnahme gilt.
- Action— Die ausstehende Wartungsmaßnahme, die auf die Ressource angewendet wird.

  Zulässige Werte:

  - ENGINE_UPDATE
  - ENGINE_UPGRADE
  - SECURITY_UPDATE
  - OS_UPDATE

  - MASTER_USER_PASSWORD_UPDATE
- AutoAppliedAfterDate— Erstes Wartungsfenster nach diesem Datum. NEXT_MAINTENANCE OPT_INwird in diesem Fall ignoriert.
- ForcedApplyDate— Wird unabhängig vom Wartungsfenster angewendet. IMMEDIATE OPT_INwird in diesem Fall ignoriert.
- OptInStatus— Ein Wert, der den Typ der Opt-in-Anfrage angibt oder eine Opt-In-Anfrage rückgängig macht. Eine Opt-in-Anfrage vom Typ IMMEDIATE kann nicht rückgängig gemacht werden.

  Zulässige Werte:

  - IMMEDIATE— Wenden Sie die Wartungsmaßnahme sofort an.
  - NEXT_MAINTENANCE— Wenden Sie die Wartungsaktion im nächsten Wartungsfenster für die Ressource an.
  - APPLY_ON— Wendet die Wartungsaktion am angegebenen Anwendungsdatum an, unabhängig vom nächsten Wartungsfenster für die Ressource.
  - UNDO_OPT_IN— Storniert alle bestehenden Anfragen NEXT_MAINTENANCE oder APPLY_ON Opt-in-Anfragen.
- CurrentApplyDate— Wird angezeigt, wenn Opt-In-Type lautet. APPLY_ON
- Description— Eine Optionsbeschreibung für die Wartungsaktion.

Termine anwenden

Jede Wartungsaktivität hat ein entsprechendes Anwendungsdatum, das Sie bei der Beschreibung der anstehenden Wartungsaktivität finden. Wenn Sie die Ausgabe der ausstehenden Wartungsmaßnahmen aus dem lesen AWS CLI, werden drei Daten aufgeführt:

- CurrentApplyDate— Das Datum, an dem die Wartungsmaßnahmen entweder sofort oder während des nächsten Wartungsfensters angewendet werden. Wenn die Wartung optional ist, kann dieser Wert Null sein.
- ForcedApplyDate— Das Datum, an dem die Wartung automatisch angewendet wird, unabhängig von Ihrem Wartungsfenster.
- AutoAppliedAfterDate— Das Datum, nach dem die Wartung während des Wartungsfensters des Clusters durchgeführt wird.

User-created Wartungsmaßnahmen

Als Amazon DocumentDBELastic-Cluster-Benutzer können Sie Aktualisierungen Ihrer Cluster-Konfigurationen initiieren.

Das primäre Cluster-Passwort wird aktualisiert

```
aws docdb-elastic update-cluster
--cluster-arn string-arn
[--admin-user-password string]
[--auth-type string-enum]
[--apply-method string-enum]
[--apply-on string-date-range]
#... other parameters of the API that follow here are not relevant for this
 configuration
```

Parameter:

- **`--cluster-arn`**— Der Amazon DocumentDB Amazon-Ressourcenname (ARN) der Ressource, auf die die Wartungsaktion angewendet wird.
- **`[--admin-user-password]`**— Das dem Admin-Benutzer zugeordnete Passwort.
- **`[--auth-type]`**— Der Authentifizierungstyp, der verwendet wird, um zu bestimmen, wo das für den Zugriff auf den Elastic Cluster verwendete Passwort abgerufen werden soll. Gültige Typen sind `PLAIN_TEXT` oder`SECRET_ARN`.
- **`[--apply-method]`**— Ein Wert, der den Typ der angewendeten Methode angibt. Zulässige Werte sind `IMMEDIATE` und `APPLY_ON`. Der Standardwert ist `IMMEDIATE`.
- **`[--apply-on]`**— Erforderlich, wenn ja`apply-method`. `APPLY_ON` Format: `yyyy/MM/dd HH:mm-yyyy/MM/dd HH:mm` (Diese Option verwendet UTC-Zeit. Die Startzeit kann zu einem beliebigen Zeitpunkt in der Zukunft zwischen mindestens 30 Minuten und maximal 14 Tagen liegen. Das Zeitfenster von Anfang bis Ende kann mindestens 30 Minuten bis maximal 8 Stunden lang sein.)

Die Ausgabe dieses Vorgangs sieht ungefähr wie folgt aus (JSON-Format):

```
{
 'ResourcePendingMaintenanceAction': {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
```

```
                'Action': 'MASTER_USER_PASSWORD_UPDATE',
                'OptInStatus': 'APPLY_ON',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            },
        ]
 }
}
```

Ändern Sie Ihre Amazon DocumentDB-Wartungsfenster

Das Wartungsfenster sollte in den Zeitraum mit der geringsten Nutzung fallen und daher unter Umständen von Zeit zu Zeit geändert werden. Ihr Elastic Cluster ist während dieser Zeit nur dann nicht verfügbar, wenn Systemänderungen (z. B. eine Änderung des Speicherbetriebs) vorgenommen werden und ein Ausfall erforderlich ist. Er ist nur für den Zeitraum nicht verfügbar, der für die Durchführung der erforderlichen Änderungen erforderlich ist.

Die Standardeinstellung ist ein 30-minütiges Fenster, das nach dem Zufallsprinzip aus einem 8-Stunden-Zeitblock für jede Amazon Web Services-Region ausgewählt wird und an einem zufälligen Wochentag stattfindet.

Informationen zum Ändern des Wartungsfensters finden Sie unterElastische Cluster-Konfigurationen ändern.

Aktualisierungen des Elastic Cluster-Betriebssystems

Elastische Amazon DocumentDB-Cluster erfordern gelegentlich Betriebssystemaktualisierungen. Amazon DocumentDB aktualisiert das Betriebssystem auf eine neuere Version, um die Datenbankleistung und die allgemeine Sicherheitslage der Kunden zu verbessern. Betriebssystemupdates ändern nicht die Cluster-Engine-Version eines Amazon DocumentDB Elastic Clusters.

Die meisten Betriebssystemupdates für Amazon DocumentDB Elastic Clusters sind optional und es gibt kein festgelegtes Datum, an dem sie angewendet werden könnten. Wenn Sie diese Updates jedoch eine Zeit lang nicht anwenden, werden sie möglicherweise irgendwann erforderlich und werden automatisch während des Wartungsfensters Ihres Clusters angewendet. Dies dient dazu, den Sicherheitsstatus Ihrer Datenbank aufrechtzuerhalten. Um unerwartete Ausfallzeiten zu vermeiden, wenden Sie Betriebssystem-Updates auf Ihren Amazon DocumentDB Elastic Cluster an, sobald diese verfügbar sind, und legen Sie Ihr Cluster-Wartungsfenster auf einen Zeitpunkt fest, der Ihren Geschäftsanforderungen entspricht.

Datenverschlüsselung im Ruhezustand für elastische Amazon DocumentDB-Cluster

Die folgenden Themen helfen Ihnen, mehr über AWS Key Management Service Verschlüsselungsschlüssel für Amazon DocumentDB Elastic Clusters zu erfahren, diese zu erstellen und zu überwachen:

Topics

- So verwenden Amazon DocumentDB Elastic Clusters Zuschüsse in AWS KMS
- Erstellen eines kundenseitig verwalteten Schlüssels
- Überwachung Ihrer Verschlüsselungsschlüssel für Amazon DocumentDB Elastic Clusters
- Weitere Informationen

Elastische Amazon DocumentDB-Cluster integrieren sich zur Schlüsselverwaltung automatisch in AWS Key Management Service (AWS KMS) und verwenden zum Schutz Ihrer Daten eine Methode, die als Umschlagverschlüsselung bekannt ist. Weitere Informationen zur Envelope-Verschlüsselung finden Sie unter Envelope-Verschlüsselung im AWS Key Management Service -Entwicklerhandbuch.

An AWS KMS key ist eine logische Darstellung eines Schlüssels. Der KMS-Schlüssel enthält Metadaten wie die Schlüssel-ID, das Erstellungsdatum, die Beschreibung und den Schlüsselstatus. Der KMS-Schlüssel enthält auch das zur Ver- und Entschlüsselung von Daten verwendete Schlüsselmaterial. Weitere Informationen über KMS-Schlüssel finden Sie unter AWS KMS keys im AWS Key Management Service Developer Guide.

Die elastischen Amazon DocumentDB-Cluster unterstützen die Verschlüsselung mit zwei Arten von Schlüsseln:

- AWS Eigene Schlüssel — Amazon DocumentDB Elastic Clusters verwenden diese Schlüssel standardmäßig, um persönlich identifizierbare Daten automatisch zu verschlüsseln. Sie können AWS eigene Schlüssel nicht anzeigen, verwalten oder verwenden oder deren Verwendung überprüfen. Sie müssen jedoch keine Maßnahmen ergreifen oder Programme zum Schutz der Schlüssel ändern, die zur Verschlüsselung Ihrer Daten verwendet werden. Weitere Informationen finden Sie unter AWS -eigene Schlüssel im AWS Key Management Service -Entwicklerhandbuch.
- Customer-managed Schlüssel — Symmetrische Schlüssel AWS KMS keys , die Sie erstellen, besitzen und verwalten. Da Sie die volle Kontrolle über diese Verschlüsselungsebene haben, können Sie beispielsweise folgende Aufgaben ausführen:

- Festlegung und Pflege wichtiger Richtlinien
- Festlegung und Aufrechterhaltung von IAM-Richtlinien und -Zuschüssen
- Aktivieren und Deaktivieren wichtiger Richtlinien
- Kryptographisches Material mit rotierendem Schlüssel
- Hinzufügen von -Tags
- Erstellen von Schlüsselaliasen
- Schlüssel für das Löschen von Schlüsseln planen

Weitere Informationen finden Sie unter Kundenverwaltete Schlüssel im AWS Key Management Service Entwicklerhandbuch.

Important

Sie müssen einen KMS-Schlüssel für die symmetrische Verschlüsselung verwenden, um Ihren Cluster zu verschlüsseln, da Amazon DocumentDB nur KMS-Schlüssel mit symmetrischer Verschlüsselung unterstützt. Verwenden Sie keinen asymmetrischen KMS-Schlüssel, um zu versuchen, die Daten in Ihren elastischen Amazon DocumentDB-Clustern zu verschlüsseln. Weitere Informationen finden Sie im AWS KMS Entwicklerhandbuch unter Asymmetrische Schlüssel. AWS Key Management Service
Wenn Amazon DocumentDB keinen Zugriff mehr auf den Verschlüsselungsschlüssel für einen Cluster erhalten kann — beispielsweise, wenn der Zugriff auf einen Schlüssel widerrufen wird — wechselt der verschlüsselte Cluster in einen Terminalzustand. In diesem Fall können Sie den Cluster nur aus einer Sicherung wiederherstellen. Für Amazon DocumentDB sind Backups immer für einen Tag aktiviert. Wenn Sie außerdem den Schlüssel für einen verschlüsselten Amazon DocumentDB-Cluster deaktivieren, verlieren Sie irgendwann den Lese- und Schreibzugriff auf diesen Cluster. Wenn Amazon DocumentDB auf einen Cluster trifft, der mit einem Schlüssel verschlüsselt ist, auf den es keinen Zugriff hat, versetzt es den Cluster in einen Terminalzustand. In diesem Fall ist der Cluster nicht länger verfügbar und der aktuelle Zustand der Datenbank kann nicht mehr wiederhergestellt werden. Um den Cluster wiederherzustellen, müssen Sie den Zugriff auf den Verschlüsselungsschlüssel für Amazon DocumentDB erneut aktivieren und dann den Cluster aus einem Backup wiederherstellen.

> **Important**
>
> Sie können den KMS-Schlüssel für einen verschlüsselten Cluster nicht mehr ändern, nachdem Sie ihn bereits erstellt haben. Stellen Sie sicher, dass Sie Ihre Anforderungen an den Verschlüsselungsschlüssel ermitteln, bevor Sie Ihren verschlüsselten Elastic Cluster erstellen.

So verwenden Amazon DocumentDB Elastic Clusters Zuschüsse in AWS KMS

Elastische Amazon DocumentDB-Cluster benötigen einen Zuschuss, um Ihren vom Kunden verwalteten Schlüssel verwenden zu können.

Wenn Sie einen Cluster erstellen, der mit einem vom Kunden verwalteten Schlüssel verschlüsselt ist, gewähren Amazon DocumentDB Elastic Clusters in Ihrem Namen einen Zuschuss, indem Sie eine `CreateGrant` Anfrage an senden. AWS KMS Grants in AWS KMS werden verwendet, um Amazon DocumentDB Elastic Clusters Zugriff auf einen KMS-Schlüssel in einem Kundenkonto zu gewähren.

Für elastische Amazon DocumentDB-Cluster ist der Zuschuss erforderlich, um Ihren vom Kunden verwalteten Schlüssel für die folgenden internen Vorgänge verwenden zu können:

- Senden Sie `DescribeKey` Anfragen an, um AWS KMS zu überprüfen, ob die symmetrische, vom Kunden verwaltete Schlüssel-ID, die Sie bei der Erstellung eines elastischen Clusters eingegeben haben, gültig ist.
- Senden Sie `GenerateDataKey` Anfragen an, AWS KMS um Datenschlüssel zu generieren, die mit Ihrem vom Kunden verwalteten Schlüssel verschlüsselt werden.
- Senden Sie `Decrypt` Anfragen AWS KMS zur Entschlüsselung der verschlüsselten Datenschlüssel, damit diese zur Verschlüsselung Ihrer Daten verwendet werden können.
- Sie können den Zugriff auf die Genehmigung jederzeit widerrufen oder den Zugriff des Services auf den vom Kunden verwalteten Schlüssel entfernen. Wenn Sie dies tun, können Amazon DocumentDB Elastic Clusters auf keine der Daten zugreifen, die mit dem vom Kunden verwalteten Schlüssel verschlüsselt wurden. Dies wirkt sich auf Vorgänge aus, die von diesen Daten abhängig sind.

Erstellen eines kundenseitig verwalteten Schlüssels

Sie können einen symmetrischen, vom Kunden verwalteten Schlüssel erstellen, indem Sie die AWS-Managementkonsole oder die AWS KMS API verwenden.

Symmetrische, vom Kunden verwaltete Schlüsselerstellung

Folgen Sie den Schritten zur Erstellung eines symmetrischen, vom Kunden verwalteten Schlüssels im AWS Key Management Service Entwicklerhandbuch.

Schlüsselrichtlinie

Schlüsselrichtlinien steuern den Zugriff auf den vom Kunden verwalteten Schlüssel. Jeder vom Kunden verwaltete Schlüssel muss über genau eine Schlüsselrichtlinie verfügen, die aussagt, wer den Schlüssel wie verwenden kann. Wenn Sie Ihren kundenseitig verwalteten Schlüssel erstellen, können Sie eine Schlüsselrichtlinie angeben. Weitere Informationen finden Sie in den Informationen zum Zugriff auf KMS-Schlüssel in der AWS Key Management Service Übersicht des AWS Key Management Service Entwicklerhandbuchs.

Um Ihren vom Kunden verwalteten Schlüssel mit Amazon DocumentDB Elastic Cluster-Ressourcen zu verwenden, müssen die folgenden API-Operationen in der Schlüsselrichtlinie zulässig sein:

- `kms:CreateGrant`: Fügt einem kundenverwalteten Schlüssel eine Erteilung hinzu. Gewährt Kontrollzugriff auf einen bestimmten KMS-Schlüssel, der den Zugriff auf die Grant-Operationen ermöglicht, die Amazon DocumentDB Elastic Clusters benötigen. Weitere Informationen zur Verwendung von Zuschüssen finden Sie AWS KMS im AWS Key Management Service Entwicklerhandbuch unter Zuschüsse.
- `kms:DescribeKey`— Stellt die vom Kunden verwalteten Schlüsseldetails bereit, damit Amazon DocumentDB Elastic Clusters den Schlüssel validieren kann.
- `kms:Decrypt`— Ermöglicht Amazon DocumentDB Elastic Clusters, den gespeicherten verschlüsselten Datenschlüssel für den Zugriff auf verschlüsselte Daten zu verwenden.
- `kms:GenerateDataKey`— Ermöglicht Amazon DocumentDB Elastic Clusters, einen verschlüsselten Datenschlüssel zu generieren und zu speichern, da der Datenschlüssel nicht sofort zum Verschlüsseln verwendet wird.

Weitere Informationen finden Sie unter Berechtigungen für AWS Dienste in wichtigen Richtlinien und Problembehandlung beim Schlüsselzugriff im AWS Key Management Service Entwicklerhandbuch.

Beschränkung des Zugriffs auf vom Kunden verwaltete Schlüssel mithilfe von IAM-Richtlinien

Zusätzlich zu den KMS-Schlüsselrichtlinien können Sie auch die KMS-Schlüsselberechtigungen in einer IAM-Richtlinie einschränken.

Sie können die IAM-Richtlinie auf verschiedene Weise strikter gestalten. Um beispielsweise zu ermöglichen, dass der vom Kunden verwaltete Schlüssel nur für Anfragen verwendet wird, die ihren Ursprung in Amazon DocumentDB Elastic Clustern haben, können Sie den `kms:ViaService` Bedingungsschlüssel mit dem `docdb-elastic.<region-name>.amazonaws.com` Wert verwenden.

Weitere Informationen finden Sie unter Benutzern in anderen Konten die Verwendung eines KMS-Schlüssels erlauben im AWS Key Management Service -Entwicklerhandbuch.

Überwachung Ihrer Verschlüsselungsschlüssel für Amazon DocumentDB Elastic Clusters

Wenn Sie einen vom Kunden verwalteten Schlüssel für Ihre Amazon DocumentDB Elastic Cluster-Ressourcen verwenden, können Sie Amazon CloudWatch Logs verwenden AWS CloudTrail , um Anfragen zu verfolgen, an die Amazon DocumentDB Elastic Clusters gesendet werden. AWS KMS

Die folgenden Beispiele sind AWS CloudTrail Ereignisse für`CreateGrant`,, und `DescribeKey` zur Überwachung von AWS KMS key Vorgängen `GenerateDataKeyWithoutPlainTextDecrypt`, die von Amazon DocumentDB Elastic Clustern aufgerufen werden, um auf Daten zuzugreifen, die mit Ihrem vom Kunden verwalteten Schlüssel verschlüsselt wurden:

CreateGrant

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws:iam::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Sampleuser01"
```

```
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2023-05-09T23:04:20Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "docdb-elastic.amazonaws.com"
    },
    "eventTime": "2023-05-09T23:55:48Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "docdb-elastic.amazonaws.com",
    "userAgent": "docdb-elastic.amazonaws.com",
    "requestParameters": {
        "retiringPrincipal": "docdb-elastic.us-east-1.amazonaws.com",
        "granteePrincipal": "docdb-elastic.us-east-1.amazonaws.com",
        "operations": [
            "Decrypt",
            "Encrypt",
            "GenerateDataKey",
            "GenerateDataKeyWithoutPlaintext",
            "ReEncryptFrom",
            "ReEncryptTo",
            "CreateGrant",
            "RetireGrant",
            "DescribeKey"
        ],
        "keyId": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
        "keyId": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "AWS Internal",
```

```
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

GenerateDataKey

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws:iam::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Sampleuser01"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2023-05-10T18:02:59Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "docdb-elastic.amazonaws.com"
    },
    "eventTime": "2023-05-10T18:03:25Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "docdb-elastic.amazonaws.com",
    "userAgent": "docdb-elastic.amazonaws.com",
```

```
    "requestParameters": {
        "keySpec": "AES_256",
        "keyId": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Decrypt

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws:iam::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Sampleuser01"
            },
            "webIdFederationData": {},
            "attributes": {
```

```
                "creationDate": "2023-05-10T18:05:49Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "docdb-elastic.amazonaws.com"
    },
    "eventTime": "2023-05-10T18:06:19Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "docdb-elastic.amazonaws.com",
    "userAgent": "docdb-elastic.amazonaws.com",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

DescribeKey

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
```

```
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws:iam::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Sampleuser01"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2023-05-09T23:04:20Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "docdb-elastic.amazonaws.com"
    },
    "eventTime": "2023-05-09T23:55:48Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "docdb-elastic.amazonaws.com",
    "userAgent": "docdb-elastic.amazonaws.com",
    "requestParameters": {
        "keyId": "alias/SampleKmsKey"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Weitere Informationen

Die folgenden Ressourcen enthalten weitere Informationen zur Verschlüsselung von Daten im Ruhezustand:

- Weitere Informationen zu AWS KMS Konzepten finden Sie unter AWS Key Management Service Grundkonzepte im AWS Key Management Service Entwicklerhandbuch.
- Weitere Informationen zur AWS KMS Sicherheit finden Sie AWS Key Management Service im AWS Key Management Service Entwicklerhandbuch unter Bewährte Sicherheitsmethoden für.

Service-linked Rollen in elastischen Clustern

Die elastischen Amazon DocumentDB-Cluster verwenden AWS Identity and Access Management (IAM) -Serviceverknüpfte Rollen. Eine serviceverknüpfte Rolle ist eine einzigartige Art von IAM-Rolle, die direkt mit den elastischen Amazon DocumentDB-Clustern verknüpft ist. Service-linked Rollen werden von Amazon DocumentDB Elastic Clusters vordefiniert und beinhalten alle Berechtigungen, die der Service benötigt, um andere AWS Services in Ihrem Namen aufzurufen.

Eine serviceverknüpfte Rolle erleichtert die Verwendung von Amazon DocumentDB Elastic Clusters, da Sie die erforderlichen Berechtigungen nicht manuell hinzufügen müssen. Amazon DocumentDB Elastic Clusters definiert die Berechtigungen seiner serviceverknüpften Rollen. Sofern nicht anders definiert, können nur Amazon DocumentDB Elastic Clusters ihre Rollen übernehmen. Die definierten Berechtigungen umfassen die Vertrauens- und Berechtigungsrichtlinie. Diese Berechtigungsrichtlinie kann keinen anderen IAM-Entitäten zugewiesen werden. Sie können die Rollen nur nach dem Löschen der zugehörigen Ressourcen löschen. Dadurch werden Ihre Amazon DocumentDB Elastic Cluster-Ressourcen geschützt, da Sie nicht versehentlich die Zugriffsberechtigung für die Ressourcen entziehen können.

Informationen zu anderen Diensten, die serviceverknüpfte Rollen unterstützen, finden Sie unter AWS Dienste, die mit IAM funktionieren. Suchen Sie nach den Diensten, die in der Spalte Rolle mit Ja markiert sind. Service-Linked Wählen Sie über einen Link Ja aus, um die Dokumentation zu einer serviceverknüpften Rolle für diesen Service anzuzeigen.

Service-linked Rollenberechtigungen für elastische Cluster

Amazon DocumentDB Elastic Clusters verwendet die serviceverknüpfte Rolle namens`AWS ServiceRoleForDocDB-Elastic`, damit Amazon DocumentDB Elastic Clusters AWS Services im Namen Ihrer Cluster aufrufen kann.

Dieser dienstgebundenen Rolle ist eine Berechtigungsrichtlinie namens `AmazonDocDB-ElasticServiceRolePolicy` zugeordnet, die ihr Berechtigungen für den Betrieb in Ihrem Konto erteilt. Die Richtlinie für Rollenberechtigungen ermöglicht es Amazon DocumentDB Elastic Clusters, die folgenden Aktionen für die angegebenen Ressourcen durchzuführen:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": [
                        "AWS/DocDB-Elastic"
                    ]
                }
            }
        }
    ]
}
```

> **Note**
>
> Sie müssen Berechtigungen konfigurieren, damit eine juristische Stelle von IAM (z. B. Benutzer, Gruppe oder Rolle) eine serviceverknüpfte Rolle erstellen, bearbeiten oder löschen kann. Wenn die folgende Fehlermeldung angezeigt wird: „Die Ressource konnte nicht erstellt werden. Überprüfen Sie, ob Sie die Berechtigung haben, eine serviceverknüpfte Rolle zu erstellen. Andernfalls warten Sie und versuchen Sie es später erneut.“ , stellen Sie sicher, dass Sie die folgenden Berechtigungen aktiviert haben:

```
{
"Action": "iam:CreateServiceLinkedRole",
    "Effect": "Allow",
    "Resource": "arn:aws:iam::*:role/aws-service-role/docdb-elastic.amazonaws.com/
AWSServiceRoleForDocDB-Elastic",
    "Condition": {
"StringLike": {
"iam:AWSServiceName":"docdb-elastic.amazonaws.com"
        }
    }
}
```

Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Service-linked Rollenberechtigungen.

Erstellen einer serviceverknüpften Rolle für Amazon DocumentDB Elastic Clusters

Sie müssen eine serviceverknüpfte Rolle nicht manuell erstellen. Wenn Sie eine DB-Instance erstellen, erstellt Amazon DocumentDB Elastic Clusters die serviceverknüpfte Rolle für Sie.

Bearbeitung einer serviceverknüpften Rolle für Amazon DocumentDB Elastic Clusters

Die elastischen Amazon DocumentDB-Cluster ermöglichen es Ihnen nicht, die AWS `ServiceRoleForDocDB-Elastic` serviceverknüpfte Rolle zu bearbeiten. Da möglicherweise verschiedene Entitäten auf die Rolle verweisen, kann der Rollenname nach dem Erstellen einer serviceverknüpften Rolle nicht mehr geändert werden. Sie können jedoch die Beschreibung der Rolle mit IAM bearbeiten. Weitere Informationen finden Sie unter Bearbeiten einer serviceverknüpften Rolle im IAM-Benutzerhandbuch.

Löschen einer serviceverknüpften Rolle für Amazon DocumentDB Elastic Clusters

Wenn Sie eine Funktion oder einen Service, für den eine serviceverknüpfte Rolle erforderlich ist, nicht mehr benötigen, löschen Sie diese Rolle. Auf diese Weise haben Sie keine ungenutzte Entität, die nicht aktiv überwacht oder verwaltet wird. Sie müssen jedoch alle Ihre -Cluster löschen, bevor Sie die serviceverknüpfte Rolle löschen können.

Bereinigen einer serviceverknüpften Rolle

Bevor Sie mit IAM eine serviceverknüpfte Rolle löschen können, müssen Sie sich zunächst vergewissern, dass die Rolle über keine aktiven Sitzungen verfügt, und alle Ressourcen entfernen, die von der Rolle verwendet werden.

Gehen Sie wie folgt vor, um zu überprüfen, ob die serviceverknüpfte Rolle eine aktive Sitzung in der IAM-Konsole hat:

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die IAM-Konsole.
2. Wählen Sie im Navigationsbereich der IAM Console Roles (Rollen) aus. Wählen Sie dann den Namen (nicht das Kontrollkästchen) der Rolle `AWS ServiceRoleForDocDB-Elastic` aus.
3. Wählen Sie auf der Seite Summary (Zusammenfassung) für die ausgewählte Rolle die Registerkarte Access Advisor (Advisor aufrufen) aus.

> **Note**
>
> Wenn Sie sich nicht sicher sind, ob Amazon DocumentDB Elastic Clusters die `AWS ServiceRoleForDocDB-Elastic` Rolle verwendet, können Sie versuchen, die Rolle zu löschen. Wenn der Service die Rolle verwendet, schlägt das Löschen fehl und Sie können sehen AWS-Regionen , wo die Rolle verwendet wird. Wenn die Rolle verwendet wird, müssen Sie warten, bis die Sitzung beendet wird, bevor Sie die Rolle löschen können. Die Sitzung für eine serviceverknüpfte Rolle können Sie nicht widerrufen.
> Wenn Sie die `AWS ServiceRoleForDocDB-Elastic` Rolle entfernen möchten, müssen Sie zuerst alle Ihre Cluster löschen.

Löschen aller Ihrer Cluster

Um einen Cluster in der Amazon DocumentDB-Konsole zu löschen:

1. Melden Sie sich bei der Amazon DocumentDB-Konsole an AWS-Managementkonsole und öffnen Sie sie.
2. Klicken Sie im Navigationsbereich auf Cluster.
3. Wählen Sie den Cluster aus, den Sie löschen möchten.
4. Klicken Sie bei Actions auf Delete.

5. Wenn Sie aufgefordert werden, den endgültigen Snapshot zu erstellen? , wählen Sie Ja oder Nein.
6. Wenn Sie im vorherigen Schritt Yes (Ja) gewählt haben, geben Sie unter Final snapshot name (Endgültiger Snapshot-Name) den Namen Ihres endgültigen DB-Snapshots ein.
7. Wählen Sie Löschen aus.

> **Note**
>
> Sie können die IAM-Konsole, die IAM-CLI oder die IAM-API verwenden, um die serviceverknüpfte Rolle `AWS ServiceRoleForDocDB-Elastic` zu löschen. Weitere Informationen finden Sie unter Löschen einer serviceverknüpften Rolle im IAM-Benutzerhandbuch.

Amazon DocumentDB-Kontingente

In diesem Thema werden die Ressourcenkontingente und Benennungsbeschränkungen für Amazon DocumentDB (mit MongoDB-Kompatibilität) beschrieben.

Für bestimmte Verwaltungsfunktionen verwendet Amazon DocumentDB Betriebstechnologie, die mit Amazon Relational Database Service (Amazon RDS) und Amazon Neptune gemeinsam genutzt wird.

Themen

- Unterstützte Instance-Typen
- Unterstützte Regionen
- Regionale Kontingente
- Kontingente für die Aggregation
- Cluster-Kontingente
- Kontingente für Instanzen
- Benennungseinschränkungen:
- TTL-Einschränkungen
- Elastische Cluster-Kontingente
- Shard-Kontingente für elastische Cluster
- CPU-, Speicher-, Verbindungs- und Cursor-Kontingente pro Shard für Elastic Cluster

Unterstützte Instance-Typen

Amazon DocumentDB unterstützt On-Demand-Instances und die folgenden Instance-Typen:

- NVMe-backed:
  - R6GD-Instanztypen: `db.r6gd.xlarge`,,,`db.r6gd.2xlarge`,`db.r6gd.4xlarge`,`db.r6gd.8xlarge`. `db.r6gd.12xlarge` `db.r6gd.16xlarge`
- RAM-optimiert:
  - R8G-Instanztypen: `db.r8g.large`,,,`db.r8g.xlarge`,,`db.r8g.2xlarge`,`db.r8g.4xlarge`,`db.r8g.8xlarge`,`db.r8g` `db.r8g.16xlarge` `db.r8g.24xlarge` `db.r8g.48xlarge`

  - R6G-Instanztypen: `db.r6g.large`,,,,`db.r6g.xlarge`,`db.r6g.2xlarge`,`db.r6g.4xlarge`. `db.r6g.8xlarge` `db.r6g.12xlarge` `db.r6g.16xlarge`
  - R5-Instanztypen: `db.r5.large`,,,`db.r5.xlarge`,`db.r5.2xlarge`,`db.r5.4xlarge`,`db.r5.8xlarge`. `db.r5.12xlarge` `db.r5.16xlarge`
  - R4-Instance-Typen: `db.r4.large`, `db.r4.2xlarge`, `db.r4.4xlarge`, `db.r4.8xlarge`, `db.r4.16xlarge`.
- Überragende Leistung:
  - T4G-Instanztypen:. `db.t4g.medium`
  - T3-Instanztypen:. `db.t3.medium`

Weitere Informationen zu den unterstützten Instance-Typen und deren Spezifikationen finden Sie unter Spezifikationen der Instanzklasse.

Unterstützte Regionen

Amazon DocumentDB ist in den folgenden Sprachen verfügbar: AWS-Regionen

Name der Region	Region	Availability Zones (Compute)
USA Ost (Ohio)	`us-east-2`	3
USA Ost (Nord-Virginia)	`us-east-1`	6
USA West (Oregon)	`us-west-2`	4
Afrika (Kapstadt)	`af-south-1`	3
Südamerika (São Paulo)	`sa-east-1`	3
Asien-Pazifik (Hongkong)	`ap-east-1`	3
Asien-Pazifik (Hyderabad)	`ap-south-2`	3
Asien-Pazifik (Malaysia)	`ap-southeast-5`	3
Asien-Pazifik (Mumbai)	`ap-south-1`	3
Asien-Pazifik (Osaka)	`ap-northeast-3`	3
Asien-Pazifik (Seoul)	`ap-northeast-2`	4
Asien-Pazifik (Singapur)	`ap-southeast-1`	3
Asien-Pazifik (Sydney)	`ap-southeast-2`	3

Name der Region	Region	Availability Zones (Compute)
Asien-Pazifik (Jakarta)	ap-southeast-3	3
Asien-Pazifik (Melbourne)	ap-southeast-4	3
Asien-Pazifik (Thailand)	ap-southeast-7	3
Asien-Pazifik (Tokio)	ap-northeast-1	3
Kanada (Zentral)	ca-central-1	3
Kanada West (Calgary)	ca-west-1	3
Region China (Peking)	cn-north-1	3
China (Ningxia)	cn-northwest-1	3
Europa (Frankfurt)	eu-central-1	3
Europa (Zürich)	eu-central-2	3
Europa (Irland)	eu-west-1	3
Europa (London)	eu-west-2	3
Europa (Milan)	eu-south-1	3
Europa (Paris)	eu-west-3	3
Europa (Spain)	eu-south-2	3
Europa (Stockholm)	eu-north-1	3
Mexiko (Zentral)	mx-central-1	3

Name der Region	Region	Availability Zones (Compute)
Naher Osten (VAE)	`me-central-1`	3
Israel (Tel Aviv)	`il-central-1`	3
AWS GovCloud (US-West)	`us-gov-west-1`	3
AWS GovCloud (US-East)	`us-gov-east-1`	3

Regionale Kontingente

Für bestimmte Verwaltungsfunktionen verwendet Amazon DocumentDB eine Betriebstechnologie, die mit Amazon Relational Database Service (Amazon RDS) gemeinsam genutzt wird. Die folgende Tabelle enthält regionale Kontingente, die von Amazon DocumentDB und Amazon RDS gemeinsam genutzt werden.

> **Note**
>
> Die oben beschriebene Amazon RDS-Technologie für gemeinsame Nutzung gilt nur für auf Amazon DocumentDB-Instances basierende Cluster. Elastische Amazon DocumentDB-Cluster nutzen die Technologie nicht gemeinsam mit Amazon RDS.

Die folgenden Kontingente gelten für auf Amazon DocumentDB-Instances basierende Cluster und gelten pro Region. AWS-Konto

Ressource	AWS Standardkontingent
Cluster	40
Cluster-Parametergruppen	50
Cluster pro Parametergruppe	100
Instanzen pro Parametergruppe	300

Ressource	AWS Standardkontingent
Ereignisabonnements	20
Instances	40
Manuelle Cluster-Snapshots	100
Read Replicas pro Cluster	15
Subnetzgruppen	50
Subnetze pro Subnetzgruppe	20
Tags pro Ressource	50
VPC-Sicherheitsgruppen pro Instance	5

Die folgenden Kontingente gelten für Amazon DocumentDB Elastic Clusters und gelten AWS-Konto pro Region.

Ressource	AWS Standardkontingent
Elastische Cluster	20
Elastische Cluster vCPU	1024
Manueller elastischer Cluster-Snapshot	20

Sie können mit Service Quotas eine Erhöhung für ein Kontingent beantragen, sofern das Kontingent anpassbar ist. Einige Anfragen werden automatisch gelöst, während andere an gesendet werden Support. Sie können den Status einer Anfrage zur Kontingenterhöhung verfolgen, die an gestellt wurde Support. Anfragen zur Erhöhung der Servicekontingente erhalten keinen bevorzugten Support. Wenn Sie eine dringende Anfrage haben, wenden Sie sich an Support. Weitere Informationen zu Service Quotas finden Sie unter What Is Service Quotas?.

Um eine Kontingenterhöhung für Amazon DocumentDB zu beantragen:

1. Öffnen Sie die Service Quotas-Konsole unter https://console.aws.amazon.com/servicequotas und melden Sie sich bei Bedarf an.
2. Wählen Sie im Navigationsbereich AWS -Services.
3. Wählen Sie Amazon DocumentDB (mit MongoDB-Kompatibilität) oder Amazon DocumentDB Elastic Cluster aus der Liste aus, oder geben Sie beides in das Suchfeld ein.
4. Wenn das Kontingent einstellbar ist, können Sie sein Optionsfeld oder seinen Namen auswählen und dann rechts oben auf der Seite die Option Request quota increase (Kontingenterhöhung beantragen) auswählen.
5. Geben Sie unter Change quota value (Kontingentwert ändern) den neuen Wert ein. Der neue Wert muss größer als der aktuelle Wert sein.
6. Wählen Sie Anfrage aus. Nachdem die Anfrage genehmigt wurde, wird Applied quota value (Angewandter Kontingentwert) für das Kontingent auf den neuen Wert eingestellt.
7. Um ausstehende oder kürzlich genehmigte Anfragen anzuzeigen, wählen Sie im Navigationsbereich die Option Dashboard . Wählen Sie für ausstehende Anfragen den Status der Anfrage, um die Anfrage zu öffnen. Der Anfangsstatus einer Anfrage ist `Pending`. Nachdem sich der Status in geändert hat`Quota requested`, sehen Sie die Fallnummer mit. Support Wählen Sie die Fallnummer, um das Ticket für Ihre Anfrage zu öffnen.

Kontingente für die Aggregation

In der folgenden Tabelle werden Aggregationsquoten in Amazon DocumentDB beschrieben.

Ressource	Kontingent
Maximale Anzahl der unterstützten Stages	500

Cluster-Kontingente

In der folgenden Tabelle werden Cluster-Kontingente beschrieben, die auf Amazon DocumentDB-Instances basieren.

Ressource	Kontingent
Cluster-Größe (Summe aller Sammlungen und Indizes)	256 TiB für Engine-Version 8.0 und höher (128 TiB für frühere Engine-Versionen)
Sammlungsgröße (die Summe aller Sammlungen darf das Cluster-Kontingent nicht überschreiten) — beinhaltet nicht die Indexgröße	32 TiB
Sammlungen pro Cluster	100 000
Datenbanken pro Cluster	100 000
Datenbankgröße (die Summe aller Datenbanken darf das Clusterkontingent nicht überschreiten)	256 TiB für Engine-Version 8.0 und höher (128 TiB für frühere Engine-Versionen)
Verschachtelungstiefe für Dokumente	200 Stufen
Dokumentengröße	16 MiB
Indexschlüsselgröße	2048 Bytes
Indizes pro Sammlung	64

Ressource	Kontingent
Schlüssel in einem zusammengesetzten Index	32
Maximale Anzahl der Schreibvorgänge in einem einzelnen Stapelbefehl	100 000
Anzahl Benutzer pro Cluster	1000

Kontingente für Instanzen

In der folgenden Tabelle werden die Amazon DocumentDB-Kontingente pro Instance beschrieben.

Amazon DocumentDB 8.0

Instance-Typ	Instanzspeicher (GiB)	Verbindungen (alle)	Cursor-Quote	Transaktionen öffnen	Verbindungen (aktiv)
T3.medium	4	1000	30	50	102
T4G.medium	4	1000	30	50	102
R5.large	16	3400	450	200	1100
R5.xlarge	32	7000	450	400	2700
R5.2xlarge	64	14200	450	800	4500
R5.4xlarge	128	28400	760	1600	4500
R5.8xlarge	256	60000	1520	3200	4500
R5.12xlarge	384	60000	2280	4800	4500
R5.16xlarge	512	60000	3040	6400	4500
R6G.large*	16	3400	450	200	1100
R6G.xlarge*	32	7000	450	400	2700
R6G.2xlarge*	64	14200	450	800	4500
R6G.4xlarge*	128	28400	760	1600	4500
R6G.8xlarge*	256	60000	1520	3200	4500

Instance-Typ	Instanzspeicher (GiB)	Verbindungen (alle)	Cursor-Quote	Transaktionen öffnen	Verbindungen (aktiv)
R6G.12xlarge*	384	60000	2280	4800	4500
R6G.16xlarge*	512	60000	3040	6400	4500
R8G.large	16	3400	450	200	1100
R8G.xlarge	32	7000	450	400	2700
R8G.2xlarge	64	14200	450	800	4500
R8G.4xlarge	128	28400	760	1600	4500
R8G.8xlarge	256	60000	1520	3200	4500
R8G.12xlarge	384	60000	2280	4800	4500
R8G.16xlarge	512	60000	3040	6400	4500
R8G.24xlarge	768	60000	4560	9600	4500
R8G.48xlarge	1536	60000	9120	9600	4500

* einschließlich R6GD

Amazon DocumentDB 5.0

Instance-Typ	Instanzspeicher (GiB)	Verbindungen (alle)	Cursor-Quote	Transaktionen öffnen	Verbindungen (aktiv)
T3.medium	4	1000	30	50	102

Instance-Typ	Instanzspeicher (GiB)	Verbindungen (alle)	Cursor-Quote	Transaktionen öffnen	Verbindungen (aktiv)
T4G.medium	4	1000	30	50	102
R5.large	16	3400	450	200	1100
R5.xlarge	32	7000	450	400	2700
R5.2xlarge	64	14200	450	800	4500
R5.4xlarge	128	28400	760	1600	4500
R5.8xlarge	256	60000	1520	3200	4500
R5.12xlarge	384	60000	2280	4800	4500
R5.16xlarge	512	60000	3040	6400	4500
R6G.large*	16	3400	450	200	1100
R6G.xlarge*	32	7000	450	400	2700
R6G.2xlarge*	64	14200	450	800	4500
R6G.4xlarge*	128	28400	760	1600	4500
R6G.8xlarge*	256	60000	1520	3200	4500
R6G.12xlarge*	384	60000	2280	4800	4500
R6G.16xlarge*	512	60000	3040	6400	4500
R8G.large	16	3400	450	200	1100
R8G.xlarge	32	7000	450	400	2700

Instance-Typ	Instanzspeicher (GiB)	Verbindungen (alle)	Cursor-Quote	Transaktionen öffnen	Verbindungen (aktiv)
R8G.2xlarge	64	14200	450	800	4500
R8G.4xlarge	128	28400	760	1600	4500
R8G.8xlarge	256	60000	1520	3200	4500
R8G.12xlarge	384	60000	2280	4800	4500
R8G.16xlarge	512	60000	3040	6400	4500
R8G.24xlarge	768	60000	4560	9600	4500
R8G.48xlarge	1536	60000	9120	9600	4500

* einschließlich R6GD

Amazon DocumentDB 4.0

Instance-Typ	Instanzspeicher (GiB)	Verbindungen (alle)	Cursor-Quote	Transaktionen öffnen	Verbindungen (aktiv)
T3.medium	4	1000	30	50	102
T4G.medium	4	1000	30	50	102
R5.large	16	3400	450	200	1100
R5.xlarge	32	7000	450	400	2700
R5.2xlarge	64	14200	450	800	4500
R5.4xlarge	128	28400	760	1600	4500

Instance-Typ	Instanzspeicher (GiB)	Verbindungen (alle)	Cursor-Quote	Transaktionen öffnen	Verbindungen (aktiv)
R5.8xlarge	256	60000	1520	3200	4500
R5.12xlarge	384	60000	2280	4800	4500
R5.16xlarge	512	60000	3040	6400	4500
R6G.large*	16	3400	450	200	1100
R6G.xlarge*	32	7000	450	400	2700
R6G.2xlarge*	64	14200	450	800	4500
R6G.4xlarge*	128	28400	760	1600	4500
R6G.8xlarge*	256	60000	1520	3200	4500
R6G.12xlarge*	384	60000	2280	4800	4500
R6G.16xlarge*	512	60000	3040	6400	4500

* einschließlich R6GD

Amazon DocumentDB 3.6

Instance-Typ	Instanzspeicher (GiB)	Verbindungen (alle)	Cursor-Quote	Transaktionen öffnen	Verbindungen (aktiv)
T3.medium	4	500	30	N/A	102
R4.large	15,25	1700	450	N/A	1100
R4.xlarge	30,5	3400	450	N/A	2700

Instance-Typ	Instanzspeicher (GiB)	Verbindungen (alle)	Cursor-Quote	Transaktionen öffnen	Verbindungen (aktiv)
R4.2xlarge	61	6800	450	N/A	4500
R4.4xlarge	122	13600	725	N/A	4500
R4.8xlarge	288	27200	1450	N/A	4500
R4.16xlarge	488	30000	2900	N/A	4500
R5.large	16	1700	450	N/A	1100
R5.xlarge	32	3500	450	N/A	2700
R5.2xlarge	64	7100	450	N/A	4500
R5.4xlarge	128	14200	760	N/A	4500
R5.8xlarge	256	28400	1520	N/A	4500
R5.12xlarge	384	30000	2280	N/A	4500
R5.16xlarge	512	30000	3040	N/A	4500

Mithilfe der folgenden CloudWatch Metriken können Sie die Grenzwerte pro Instanz überwachen und Warnmeldungen einleiten. Weitere Informationen zu Amazon CloudWatch DocumentDB-Metriken finden Sie unter Überwachung von Amazon DocumentDB mit CloudWatch.

Ressource	CloudWatch Quoten-Metrik	CloudWatch Nutzungsmetrik (maximal 1 Minute)	CloudWatch Metrik zur Nutzung
Instanzspeicher	-	-	FreeableMemory
Verbindungen (alle)	DatabaseConnectionsLimit	DatabaseConnectionsMax	DatabaseConnections
Cursor	DatabaseCursorsLimit	DatabaseCursorsMax	DatabaseCursors

Ressource	CloudWatch Quoten-Metrik	CloudWatch Nutzungsmetrik (maximal 1 Minute)	CloudWatch Metrik zur Nutzung
Transaktionen	TransactionsOpenLimit	TransactionsOpenMax	TransactionsOpen

Benennungseinschränkungen:

In der folgenden Tabelle werden die Benennungsbeschränkungen in Amazon DocumentDB beschrieben.

Ressource	Standardkontingent
Cluster-Kennung	• Die Länge beträgt [1—63] Buchstaben, Zahlen oder Bindestriche. • Muss mit einem Buchstaben beginnen. • Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten. • Muss für alle Cluster (in Amazon RDS, Amazon Neptune und Amazon DocumentDB) pro Konto und Region eindeutig sein. AWS
Instance-ID	• Die Länge beträgt [1—63] Buchstaben, Zahlen oder Bindestriche • Muss mit einem Buchstaben beginnen • Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten • Muss für alle Instances (in Amazon RDS, Amazon Neptune und Amazon DocumentDB) pro

Ressource	Standardkontingent
	Konto und Region eindeutig sein. AWS
Name der Sammlung: `<col>`	Die Länge beträgt [1—255] Zeichen (Amazon DocumentDB 5.0 und höher). Die Länge beträgt [1—57] Zeichen (Amazon DocumentDB 4.0 und früher).
Datenbankname: `<db>`	Die Länge beträgt [1—63] Zeichen.
Vollständig qualifizierter Sammlungsname: `<db>.<col>`	Die Länge beträgt [3—255] Zeichen (Amazon DocumentDB 5.0 und höher). Die Länge beträgt [3—120] Zeichen (Amazon DocumentDB 4.0 und früher).
Vollständig qualifizierter Indexname: `<db>.<col>$<index>`	Die Länge beträgt [6—511] Zeichen (Amazon DocumentDB 5.0 und höher). Die Länge beträgt [6—127] Zeichen (Amazon DocumentDB 4.0 und früher).
Indexname	Die Länge beträgt [1—255] Zeichen (Amazon DocumentDB 5.0 und höher). Die Länge beträgt [3—63] Zeichen für die Kombination `<col>$<index>` (Amazon DocumentDB 4.0 und früher).

Ressource	Standardkontingent
Primäres Passwort	• Länge beträgt [8–100] druckbare ASCII-Zeichen. • Es können alle druckbaren ASCII-Zeichen mit Ausnahme der folgenden verwendet werden: • **/** (Schrägstrich) • **"** (doppeltes Anführung szeichen) • **@** ('At'-Symbol)
Primärer Benutzername	• Die Länge beträgt [1 bis 63] alphanumerische Zeichen. • Muss mit einem Buchstaben beginnen. • Darf kein Wort sein, das von der Datenbank-Engine reserviert ist.
Parametergruppenname	• Die Länge muss [1 bis 255] alphanumerische Zeichen betragen. • Muss mit einem Buchstaben beginnen. • Darf nicht mit einem Bindestri ch enden oder zwei aufeinand erfolgende Bindestriche enthalten.

TTL-Einschränkungen

Es kann nicht garantiert werden, dass Löschoperationen über einem TTL-Index innerhalb eines bestimmten Zeitraums abgeschlossen und mit höchster Priorität behandelt werden. Faktoren wie

Ressourcenauslastung der Instance, Dokumentgröße und Gesamtdurchsatz können Einfluss auf die Dauer von TTL-Löschoperationen haben.

Elastische Cluster-Kontingente

In der folgenden Tabelle werden die maximalen Kontingente für elastische Amazon DocumentDB-Cluster beschrieben.

Ressource	Kontingent
Elastische Cluster pro Region	20
vCPU, summiert über alle elastischen Cluster pro Region	1024
Manuelle Cluster-Snapshots pro Region	20
Shards pro Cluster	32
Speicher pro Cluster (wenn die Daten gleichmäßig per Shard-Key verteilt werden)	4 PiB
Verbindungen zum Cluster	Der niedrigere Wert von entweder 300.000 oder die Anzahl der Shards x das Verbindungskontingent, das der vCPU pro Shard zugeordnet ist
UnSharded Größe der Sammlung	32 TiB
Größe der geteilten Sammlung (wenn die Daten gleichmäßig per Shard-Key verteilt werden)	1 PB
Datenbanken pro Cluster	10.000
UnSharded Sammlungen pro Cluster	100 000
Geteilte Sammlungen pro Cluster	1000
Benutzer pro Cluster	100

Ressource	Kontingent
Schreibt in einem einzigen Batch-Befehl	100 000
Indizes pro Sammlung	64
Verschachtelungstiefe für Dokumente	100 Ebenen
Dokumentengröße	16 MB
Indexschlüsselgröße	2048 Bytes
Schlüssel in einem zusammengesetzten Index	32

Shard-Kontingente für elastische Cluster

In der folgenden Tabelle werden die maximalen Shard-Kontingente für elastische Amazon DocumentDB-Cluster beschrieben.

Ressource	Kontingent
vCPU pro Shard-Instance	64
Instanzen pro Shard	16
Speicherplatz pro Shard	128 TiB
Speicherplatz pro Sammlung pro Shard	32 TiB

CPU-, Speicher-, Verbindungs- und Cursor-Kontingente pro Shard für Elastic Cluster

In der folgenden Tabelle werden die maximalen CPU-, Speicher-, Verbindungs- und Cursorgrenzen in Amazon DocumentDB Elastic Cluster-Shards beschrieben.

vCPUs pro Shard	Instanzspeicher (GiB)	Verbindungskontingent	Cursor-Quote
2	16	1700	450
4	32	3500	450
8	64	7100	450
16	128	14200	760
32	256	28400	1520
48	384	30000	2280
64	512	30000	3040

Problembehandlung bei Amazon DocumentDB

Die folgenden Abschnitte enthalten Informationen zur Behebung von Problemen, die bei der Verwendung von Amazon DocumentDB (mit MongoDB-Kompatibilität) auftreten können.

Themen

- Fehlerbehebung bei Verbindungsproblemen
- Problembehandlung bei Indizes
- Problembehandlung bei Leistung und Ressourcenauslastung
- Müllabfuhr in Amazon DocumentDB
- Fehlercodes

Fehlerbehebung bei Verbindungsproblemen

Haben Sie Probleme beim Herstellen einer Verbindung? Im Folgenden finden Sie einige häufig auftretende Szenarien und wie Sie sie lösen können.

Topics

- Es kann keine Verbindung zu einem Amazon DocumentDB-Endpunkt hergestellt werden
- Testen einer Verbindung zu einer Amazon DocumentDB-Instance
- Verbindung zu einem ungültigen Endpunkt herstellen
- Die Treiberkonfiguration wirkt sich auf die Anzahl der Verbindungen aus

Es kann keine Verbindung zu einem Amazon DocumentDB-Endpunkt hergestellt werden

Wenn Sie versuchen, eine Verbindung zu Amazon DocumentDB herzustellen, ist die folgende eine der häufigsten Fehlermeldungen, die Sie möglicherweise erhalten.

```
connecting to: mongodb://docdb-2018-11-08-21-47-27.cluster-ccuszbx3pn5e.us-east-
1.docdb.amazonaws.com:27017/
2018-11-14T14:33:46.451-0800 W NETWORK [thread1] Failed to connect to
```

```
172.31.91.193:27017 after 5000ms milliseconds, giving up.
2018-11-14T14:33:46.452-0800 E QUERY [thread1] Error: couldn't connect to server
docdb-2018-11-08-21-47-27.cluster-ccuszbx3pn5e.us-east-1.docdb.amazonaws.com:27017,
connection attempt failed :
connect@src/mongo/shell/mongo.js:237:13
@(connect):1:6
exception: connect failed
```

Diese Fehlermeldung bedeutet in der Regel, dass Ihr Client (in diesem Beispiel die Mongo-Shell) nicht auf den Amazon DocumentDB-Endpunkt zugreifen kann. Dies kann aus verschiedenen Gründen der Fall sein:

Themen

- Verbindung von öffentlichen Endpunkten aus herstellen
- Cross-Region Verbindungen
- Verbindung von verschiedenen Amazon VPCs aus herstellen
- Die Sicherheitsgruppe blockiert eingehende Verbindungen
- Problem mit den Leseeinstellungen für den Java Mongo-Treiber

Verbindung von öffentlichen Endpunkten aus herstellen

Sie versuchen, direkt von Ihrem Laptop oder Ihrem lokalen Entwicklungscomputer aus eine Verbindung zu einem Amazon DocumentDB-Cluster herzustellen.

Der Versuch, direkt von einem öffentlichen Endpunkt, wie Ihrem Laptop oder Ihrem lokalen Entwicklungscomputer, eine Verbindung zu einem Amazon DocumentDB-Cluster herzustellen, schlägt fehl. Amazon DocumentDB ist nur für Virtual Private Cloud (VPC) verfügbar und unterstützt derzeit keine öffentlichen Endpunkte. Daher können Sie von Ihrem Laptop oder Ihrer lokalen Entwicklungsumgebung außerhalb Ihrer VPC aus keine direkte Verbindung zu Ihrem Amazon DocumentDB-Cluster herstellen.

Um von außerhalb einer Amazon VPC eine Verbindung zu einem Amazon DocumentDB-Cluster herzustellen, können Sie einen SSH-Tunnel verwenden. Weitere Informationen finden Sie unter Verbindung zu einem Amazon DocumentDB-Cluster von außerhalb einer Amazon VPC herstellen. Wenn sich Ihre Entwicklungsumgebung in einer anderen Amazon VPC befindet, können Sie außerdem VPC Peering verwenden und von einer anderen Amazon VPC in derselben Region oder einer anderen Region aus eine Verbindung zu Ihrem Amazon DocumentDB-Cluster herstellen.

Cross-Region Verbindungen

Sie versuchen, eine Verbindung zu einem Amazon DocumentDB-Cluster in einer anderen Region herzustellen.

Wenn Sie versuchen, von einer Amazon EC2-Instance aus einer anderen Region als der Cluster-Region eine Verbindung zu einem Amazon DocumentDB-Cluster herzustellen — beispielsweise wenn Sie versuchen, eine Verbindung zu einem Cluster in der Region USA Ost (Nord-Virginia) (us-east-1) von der Region US West (Oregon) (us-west-2) herzustellen —, schlägt die Verbindung fehl.

Führen Sie den folgenden Befehl aus, um die Region Ihres Amazon DocumentDB-Clusters zu überprüfen. Die Region ist im Endpunkt.

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].Endpoint'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
[
    "sample-cluster.node.us-east-1.docdb.amazonaws.com"
]
```

Um die Region Ihrer EC2-Instance zu überprüfen, führen Sie den folgenden Befehl aus.

```
 aws ec2 describe-instances \
     --query 'Reservations[*].Instances[*].Placement.AvailabilityZone'
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus.

```
[
    [
        "us-east-1a"
    ]
]
```

Verbindung von verschiedenen Amazon VPCs aus herstellen

Sie versuchen, eine Verbindung zu einem Amazon DocumentDB-Cluster von einer VPC aus herzustellen, die sich von der Amazon VPC unterscheidet, auf der Ihr Cluster bereitgestellt wird.

Wenn sich sowohl Ihr Amazon DocumentDB-Cluster als auch Ihre Amazon EC2-Instance in derselben AWS-Region, aber nicht in derselben Amazon VPC befinden, können Sie keine direkte Verbindung zu Ihrem Amazon DocumentDB-Cluster herstellen, es sei denn, VPC Peering ist zwischen den beiden Amazon VPCs aktiviert.

Führen Sie den folgenden Befehl aus, um die Amazon VPC Ihrer Amazon DocumentDB-Instance zu überprüfen.

```
aws docdb describe-db-instances \
   --db-instance-identifier sample-instance \
   --query 'DBInstances[*].DBSubnetGroup.VpcId'
```

Führen Sie den folgenden Befehl aus, um die Amazon VPC Ihrer Amazon EC2-Instance zu überprüfen.

```
aws ec2 describe-instances \
   --query 'Reservations[*].Instances[*].VpcId'
```

Die Sicherheitsgruppe blockiert eingehende Verbindungen

Sie versuchen, eine Verbindung zu einem Amazon DocumentDB-Cluster herzustellen, und die Sicherheitsgruppe des Clusters lässt keine eingehenden Verbindungen über den Port des Clusters zu (Standardport: 27017).

Angenommen, Ihr Amazon DocumentDB-Cluster und Ihre Amazon EC2-Instance befinden sich beide in derselben Region und Amazon VPC und verwenden dieselbe Amazon VPC-Sicherheitsgruppe. Wenn Sie keine Verbindung zu Ihrem Amazon DocumentDB-Cluster herstellen können, liegt die wahrscheinliche Ursache darin, dass Ihre Sicherheitsgruppe (d. h. die Firewall) für Ihren Cluster keine eingehenden Verbindungen auf dem Port zulässt, den Sie für Ihren Amazon DocumentDB-Cluster ausgewählt haben (Standardport ist 27017).

Führen Sie den folgenden Befehl aus, um den Port für Ihren Amazon DocumentDB-Cluster zu überprüfen.

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[DBClusterIdentifier,Port]'
```

Führen Sie den folgenden Befehl aus, um Ihre Amazon DocumentDB-Sicherheitsgruppe für Ihren Cluster abzurufen.

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[VpcSecurityGroups[*],VpcSecurityGroupId]'
```

Um die Regeln für eingehenden Verkehr für Ihre Sicherheitsgruppe zu überprüfen, lesen Sie die folgenden Themen in der Amazon EC2-Dokumentation:

- Autorisierung des eingehenden Datenverkehrs für Ihre Linux-Instances
- Autorisierung des eingehenden Datenverkehrs für Ihre Windows-Instances

Problem mit den Leseeinstellungen für den Java Mongo-Treiber

Die Leseeinstellungen des Clients werden nicht berücksichtigt und einige Clients können nach einem Failover nicht in Amazon DocumentDB schreiben, es sei denn, sie starten neu.

Dieses Problem, das erstmals in Java Mongo Driver 3.7.x entdeckt wurde, tritt auf, wenn ein Client mithilfe der Methode `MongoClientSettings` und insbesondere beim Verketten der Methode eine Verbindung zu Amazon DocumentDB herstellt. `applyToClusterSettings` Die MongoClient Cluster-Einstellungen können mit verschiedenen Methoden wie, und definiert werden. `hosts()` `requiredReplicaSetName()` `mode()`

Wenn der Client in der `hosts()` Methode nur einen Host angibt, wird der Modus auf `ClusterConnectionMode.SINGLE` statt gesetzt. `ClusterConnectionMode.MULTIPLE` Dies führt dazu, dass der Client die Lesepräferenz ignoriert und nur eine Verbindung zu dem in `hosts()` konfigurierten Server herstellt. Selbst wenn die Client-Einstellungen wie unten beschrieben initialisiert werden, würden alle Lesevorgänge immer noch an die primäre statt an die sekundäre gesendet.

```
final ServerAddress serverAddress0 = new ServerAddress("cluster-endpoint", 27317));
    final MongoCredential credential = MongoCredential.createCredential("xxx",
            "admin", "xxxx".toCharArray());
    final MongoClientSettings settings = MongoClientSettings.builder()
            .credential(credential)
            .readPreference(ReadPreference.secondaryPreferred())
            .retryWrites(false)
            .applyToSslSettings(builder -> builder
                    .enabled(false))
            .applyToClusterSettings(builder -> builder.hosts(
                            Arrays.asList(serverAddress0
                            ))
                    .requiredReplicaSetName("rs0"))
```

```
            .build();
    MongoClient mongoClient = MongoClients.create(settings);
```

Fall eines Failovers

Bei Verwendung der oben genannten Client-Verbindungseinstellungen würde der Client bei einem Failover und einer verzögerten DNS-Datensatzaktualisierung für den Cluster-Writer-Endpunkt trotzdem versuchen, Schreibvorgänge an den alten Writer (jetzt Reader nach Failover) auszugeben. Dies führt zu einem serverseitigen Fehler (nicht primär), der vom Java-Treiber nicht angemessen behandelt wird (dieser Fehler wird noch untersucht). So kann der Client beispielsweise so lange in einem schlechten Zustand bleiben, bis der Anwendungsserver neu gestartet wird.

Dafür gibt es zwei Workarounds:

- Bei Clients, die über eine Verbindungszeichenfolge eine Verbindung zu Amazon DocumentDB herstellen, tritt dieses Problem nicht auf, da `ClusterConnectionMode` es `MULTIPLE` bei der Einstellung der Lesepräferenz auf gesetzt wird.

  ```
  MongoClientURI mongoClientURI = new MongoClientURI("mongodb://usr:pass:cluster-
  endpoint:27317/test?ssl=false&replicaSet=rs0&readpreference=secondaryPreferred");
  MongoClient mongoClient = MongoClients.create(mongoClientURI.getURI());
  ```

  Oder verwenden Sie den `MongoClientSettings` Builder mit der `applyConnectionString` Methode.

  ```
  final MongoClientSettings settings = MongoClientSettings.builder()
          .credential(credential)
          .applyConnectionString(new ConnectionString("usr:pass:cluster-endpoint:27317/
  test?ssl=false&replicaSet=rs0&readpreference=secondaryPreferred"))
          .retryWrites(false)
          .applyToSslSettings(builder # builder
                  .enabled(false))
          .build();
  MongoClient mongoClient = MongoClients.create(settings);
  ```

- Explizit `ClusterConnectionMode` auf gesetzt`MULTIPLE`. Dies wird nur benötigt, wenn Sie `applyToClusterSettings` und verwenden`hosts().size() == 1`.

  ```
  final ServerAddress serverAddress0 = new ServerAddress("cluster-endpoint", 27317));
  final MongoCredential credential = MongoCredential.createCredential("xxx","admin",
   "xxxx".toCharArray());
  ```

```
final MongoClientSettings settings = MongoClientSettings.builder()
    .credential(credential)
    .readPreference(ReadPreference.secondaryPreferred())
    .retryWrites(false)
    .applyToSslSettings(builder # builder
    .enabled(false))
    .applyToClusterSettings(builder # builder
                .hosts(Arrays.asList(serverAddress0))
                .requiredReplicaSetName("rs0"))
                .mode(ClusterConnectionMode.MULTIPLE))
    .build();
MongoClient mongoClient = MongoClients.create(settings);
```

Testen einer Verbindung zu einer Amazon DocumentDB-Instance

Sie können Ihre Verbindung zu einem Cluster mit gängigen Linux- oder Windows-Tools testen.

Testen Sie die Verbindung über einen Linux- oder Unix-Terminal, indem Sie folgendes eingeben (ersetzen Sie `cluster-endpoint` durch den Endpunkt und ersetzen Sie `port` durch den Port Ihrer Instance).

```
nc -zv cluster-endpoint port
```

Der folgende Code ist ein Beispiel für eine Beispieloperation und den Rückgabewert:

```
nc -zv docdbTest.d4c7nm7stsfc0.us-west-2.docdb.amazonaws.com 27017

Connection to docdbTest.d4c7nm7stsfc0.us-west-2.docdb.amazonaws.com 27017 port [tcp/*]
 succeeded!
```

Verbindung zu einem ungültigen Endpunkt herstellen

Wenn Sie eine Verbindung zu einem Amazon DocumentDB-Cluster herstellen und einen Cluster-Endpunkt verwenden, der nicht gültig ist, wird ein Fehler ähnlich dem folgenden angezeigt.

```
mongo --ssl \
   --host sample-cluster.node.us-east-1.docdb.amazonaws.com:27017 \
   --sslCAFile global-bundle.pem \
   --username <user-name> \
   --password <password>
```

Das Ergebnis sieht folgendermaßen aus:

```
MongoDB shell version v3.6
connecting to: mongodb://sample-cluster.node.us-east-1.docdb.amazonaws.com:27017/
2018-11-14T17:21:18.516-0800 I NETWORK [thread1] getaddrinfo("sample-cluster.node.us-
east-1.docdb.amazonaws.com") failed:
nodename nor servname provided, or not known 2018-11-14T17:21:18.537-0800 E QUERY
 [thread1] Error: couldn't initialize
connection to host sample-cluster.node.us-east-1.docdb.amazonaws.com, address is
 invalid :
connect@src/mongo/shell/mongo.js:237:13@(connect):1:6
exception: connect failed
```

Um den gültigen Endpunkt für einen Cluster abzurufen, führen Sie den folgenden Befehl aus:

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[Endpoint,Port]'
```

Um den gültigen Endpunkt für eine Instance abzurufen, führen Sie den folgenden Befehl aus:

```
aws docdb describe-db-instances \
   --db-instance-identifier sample-instance \
   --query 'DBInstances[*].[Endpoint.Address,Endpoint.Port]'
```

Weitere Informationen finden Sie unter Grundlegendes zu Amazon DocumentDB DocumentDB-Endpunkten.

Die Treiberkonfiguration wirkt sich auf die Anzahl der Verbindungen aus

Wenn Sie den Client-Treiber verwenden, um eine Verbindung zu einem Amazon DocumentDB-Cluster herzustellen, ist es wichtig, den `maxPoolSize` Konfigurationsparameter zu berücksichtigen. Die `maxPoolSize` Einstellung bestimmt die maximale Anzahl von Verbindungen, die der Client-Treiber in seinem Verbindungspool aufrechterhalten wird.

Problembehandlung bei Indizes

Die folgenden Themen erläutern, was zu tun ist, wenn Ihr Index oder Ihr Indexaufbau im Hintergrund fehlschlägt.

Themen

- Die Indexerstellung schlägt fehl
- Latenzprobleme und Fehlschläge bei der Erstellung des Hintergrundindexes
- Aufblähung des Datenbankindex

Die Indexerstellung schlägt fehl

Amazon DocumentDB verwendet den lokalen Speicher auf einer Instance als Teil des Indexerstellungsprozesses. Sie können diese Festplattennutzung mithilfe der FreeLocalStorage CloudWatch Metrik () `CloudWatch -> Metrics -> DocDB -> Instance Metrics` überwachen. Wenn ein Indexaufbau den gesamten lokalen Festplattenspeicherplatz verbraucht und fehlschlägt, erhalten Sie eine Fehlermeldung. Bei der Migration von Daten zu Amazon DocumentDB empfehlen wir Ihnen, zuerst Indizes zu erstellen und dann die Daten einzufügen. Weitere Informationen zu Migrationsstrategien und zur Erstellung von Indizes finden Sie Migration und Aktualisierung von Amazon DocumentDB in der Amazon DocumentDB-Dokumentation und im Blog: Migrieren Sie mit der Offline-Methode von MongoDB zu Amazon DocumentDB.

Wenn beim Erstellen von Indizes auf einem vorhandenen Cluster die Indexerstellung länger als erwartet dauert oder fehlschlägt, skalieren Sie die Instance hoch, um den Index zu erstellen, und skalieren Sie dann nach der Erstellung des Indexes wieder nach unten. Amazon DocumentDB ermöglicht es Ihnen, Instance-Größen innerhalb von Minuten schnell zu skalieren, indem Sie die AWS-Managementkonsole oder verwenden. AWS CLI Weitere Informationen finden Sie unter Instanzklassen verwalten. Bei der sekundengenauen Instance-Preisgestaltung zahlen Sie auf die Sekunde genau nur für die Ressourcen, die Sie verwenden.

Latenzprobleme und Fehlschläge bei der Erstellung des Hintergrundindexes

Die Erstellung von Indizes im Hintergrund in Amazon DocumentDB beginnt erst, wenn alle Abfragen auf der primären Instance, die vor der Initiierung der Indexerstellung gestartet wurden, die Ausführung abgeschlossen haben. Wenn eine Abfrage lange läuft, werden die Indexerstellungen im Hintergrund blockiert, bis die Abfrage abgeschlossen ist. Daher kann es länger als erwartet dauern, bis die Abfrage abgeschlossen ist. Dies gilt auch dann, wenn Sammlungen leer sind.

Indexerstellungen im Vordergrund weisen nicht das gleiche Blockierungsverhalten auf. Stattdessen wird die Sammlung bei Indexerstellungen im Vordergrund exklusiv gesperrt, bis die Indexerstellung abgeschlossen ist. Verwenden Sie Vordergrundindex-Builds, um Indizes für eine leere Sammlung zu erstellen und das Blockieren lang andauernder Abfragen zu vermeiden.

> **Note**
>
> Amazon DocumentDB erlaubt zu einem bestimmten Zeitpunkt nur die Erstellung eines Hintergrundindexes für eine Sammlung. Wenn DDL (Data Definition Language) Operationen wie `createIndex()` oder `dropIndex()` während eines Indexaufbaus im HIntergrund in derselben Sammlung auftreten, schlägt der Aufbau des Hintergrundindex fehl.

Aufblähung des Datenbankindex

Amazon DocumentDB verwendet Multi-Version Concurrency Control (MVCC), um gleichzeitige Transaktionen zu verwalten. Wenn Dokumente gelöscht oder aktualisiert werden, verbleiben ihre früheren Versionen als „tote“ Versionen in Sammlungen und Indizes. Der Garbage Collection-Prozess gibt automatisch Speicherplatz aus diesen toten Versionen für zukünftige Operationen zurück.

Eine Aufblähung des Indexes tritt auf, wenn die Indizes einer Sammlung aufgrund der Anhäufung von toten oder veralteten Indexeinträgen oder einer Fragmentierung innerhalb der Seiten größer werden. Der angegebene Prozentsatz steht für die Menge an Indexplatz, die von zukünftigen Indexeinträgen genutzt werden kann. Diese Überlastung verbraucht Speicherplatz sowohl im Puffercache als auch im Speicher. Wenn Sie das Aufblähen entfernen möchten, müssen Sie die Indizes neu erstellen.

Example Beispiel

Führen Sie den folgenden Befehl aus, um ungenutzten Speicherplatz für Ihren Index zu ermitteln:

```
db.coll.aggregate({$indexStats:{}});
```

Das Ergebnis sieht ungefähr so aus:

```
{
    "name" : "_id_",
    "key" : {
        "_id" : 1
    },
    "host" : "devbox-test.localhost.a2z.com:27317",
    "size" : NumberLong(827392),
    "accesses" : {
        "ops" : NumberLong(40000),
        "docsRead" : NumberLong(46049),
```

```
        "since" : ISODate("2025-04-03T21:44:51.251Z")
    },
    "cacheStats" : {
        "blksRead" : NumberLong(264),
        "blksHit" : NumberLong(140190),
        "hitRatio" : 99.8121
    },
    "unusedStorageSize" : {
        "unusedSizeBytes" : 409600,
        "unusedSizePercent" : 49.51
    }
}
```

Mit dem `reIndex` Befehl, der einen Scan der gesamten Sammlung erfordert, können Sie Indizes ohne Ausfallzeiten neu erstellen. Siehe Indexverwaltung mithilfe von `Reindex`.

Problembehandlung bei Leistung und Ressourcenauslastung

Dieser Abschnitt enthält Fragen und Lösungen für häufig auftretende Diagnoseprobleme bei Amazon DocumentDB-Bereitstellungen. Die bereitgestellten Beispiele verwenden die mongo-Shell und gelten für eine einzelne Instance. Informationen zum Suchen nach einem Instance-Endpunkt finden Sie unter Grundlegendes zu Amazon DocumentDB DocumentDB-Endpunkten.

Themen

- Wie ermittle ich die Anzahl der Einfüge-, Aktualisierungs- und Löschvorgänge, die für meine Sammlung über die Mongo-API ausgeführt wurden?
- Wie analysiere ich die Cache-Leistung?
- Wie finde und beende ich langsame und blockierte Abfragen?
- Wie kann ich einen Abfrageplan einsehen und eine Abfrage optimieren?
- Wie kann ich einen Abfrageplan in elastischen Clustern sehen?
- Wie liste ich alle laufenden Operationen für eine Instance auf?
- Woher weiß ich, wann eine Anfrage Fortschritte macht?
- Wie stelle ich fest, warum ein System plötzlich langsam läuft?
- Wie ermittle ich die Ursache für eine hohe CPU-Auslastung auf einer oder mehreren Cluster-Instances?
- Wie ermittle ich die offenen Cursor auf einer Instance?

- Wie ermittle ich die aktuelle Version der Amazon DocumentDB-Engine?
- Wie analysiere ich die Indexnutzung und identifiziere ungenutzte Indizes?
- Wie identifiziere ich fehlende Indizes?
- Wie stelle ich fest, dass die Datenbanksammlung überfüllt ist?
- Zusammenfassung nützlicher Abfragen

Wie ermittle ich die Anzahl der Einfüge-, Aktualisierungs- und Löschvorgänge, die für meine Sammlung über die Mongo-API ausgeführt wurden?

Um die Anzahl der Einfüge-, Aktualisierungs- und Löschvorgänge anzuzeigen, die für eine bestimmte Sammlung ausgeführt wurden, führen Sie den folgenden Befehl für diese Sammlung aus:

```
db.collection.stats()
```

Die Ausgabe dieses Befehls beschreibt in seinem `opCounters` Feld Folgendes:

- num DocsIns — Die Anzahl der Dokumente, die in diese Sammlung eingefügt wurden. Dazu gehören Dokumente, die mit den `insertMany` Befehlen `insert` und eingefügt wurden, sowie Dokumente, die durch einen Upsert eingefügt wurden.
- num DocsUpd — Die Anzahl der Dokumente, die in dieser Sammlung aktualisiert wurden. Dies schließt Dokumente ein, die mit den `findAndModify` Befehlen `update` und aktualisiert wurden.
- num DocsDel — Die Anzahl der Dokumente, die aus dieser Sammlung gelöscht wurden. Dies schließt Dokumente ein`deleteOne`, die mit den `findAndModify` Befehlen`deleteMany`,`remove`, und gelöscht wurden.
- LastReset — Der Zeitpunkt, zu dem diese Zähler das letzte Mal zurückgesetzt wurden. Die von diesem Befehl bereitgestellten Statistiken werden beim Clustern oder bei starting/stopping der Skalierung up/down der Instanz zurückgesetzt.

Ein Beispiel für die Ausgabe von running `db.collection.stats()` ist unten dargestellt.

```
{
    "ns" : "db.test",
    "count" : ...,
    "size" : ...,
```

```
    "avgObjSize" : ...,
    "storageSize" : ...,
    "capped" : false,
    "nindexes" : ...,
    "totalIndexSize" : ...,
    "indexSizes" : {
        "_id_" : ...,
        "x_1" : ...
    },
    "collScans" : ...,
    "idxScans" : ...,
    "opCounter" : {
        "numDocsIns" : ...,
        "numDocsUpd" : ...,
        "numDocsDel" : ...
    },
    "cacheStats" : {
        "collBlksHit" : ...,
        "collBlksRead" : ..,
        "collHitRatio" : ...,
        "idxBlksHit" : ...,
        "idxBlksRead" : ...,
        "idxHitRatio" : ...
    },
    "lastReset" : "2022-09-02 19:41:40.471473+00",
    "ok" : 1,
    "operationTime" : Timestamp(1662159707, 1)
}
```

Dieser Statistikbefehl sollte verwendet werden, wenn sammlungsspezifische Zähler für Einfüge-, Aktualisierungs- und Löschvorgänge über die Mongo-API angezeigt werden. Eine andere Möglichkeit, sammlungsspezifische Betriebszähler anzuzeigen, besteht darin, das DML-Auditing zu aktivieren. Die Anzahl der Einfüge-, Aktualisierungs- und Löschvorgänge für alle Sammlungen in Zeitintervallen von einer Minute kann unter angezeigt werden. Überwachung von Amazon DocumentDB mit CloudWatch

Wie analysiere ich die Cache-Leistung?

Die Analyse der Cache-Leistung kann Aufschluss über die Effizienz des Datenabrufs und die Systemleistung geben. Sie basiert darauf, wie viele Daten von der Festplatte gelesen werden und wie viele Daten vom Cache gelesen werden. Wir stellen Cache-Statistiken zur Anzahl der Cache-Treffer (aus dem Cache gelesene Daten) und Cache-Fehlschläge (Daten, die nicht im Cache gefunden

und von der Festplatte gelesen wurden) zur Verfügung, um einen Einblick in die Cache-Leistung zu geben. Die Cache-Statistiken für eine bestimmte Sammlung können gefunden werden, indem Sie den folgenden Befehl für diese Sammlung ausführen:

```
db.collection.stats()
```

Die Werte in dem `cacheStats` Feld in der Ausgabe dieses Befehls liefern Cachestatistiken für die Sammlung sowie die gesamten Cachestatistiken für die Indizes, die für die Sammlung erstellt wurden. Diese Statistiken sind unten aufgeführt:

- **`collBlksHit`**- Die Anzahl der Blöcke, die bei Vorgängen mit dieser Sammlung aus dem Cache gelesen wurden.
- **`collBlksRead`**- Die Anzahl der Blöcke, die bei Vorgängen mit dieser Sammlung von der Festplatte gelesen wurden (Cache-Fehler).
- **`collHitRatio`**- Die Cache-Trefferquote für diese Sammlung (`100 * [collBlksHit / (collBlksHit + collBlksRead)]`).
- **`idxBlksHit`**- Die Anzahl der aus dem Cache gelesenen Blöcke für jeden Index, der für diese Sammlung erstellt wurde.
- **`idxBlksRead`**- Die Anzahl der von der Festplatte gelesenen Blöcke (Cache-Fehler) für jeden Index, der für diese Sammlung erstellt wurde.
- **`idxHitRatio`**- Die Cache-Trefferquote für die in dieser Sammlung erstellten Indizes (`100 * [idxBlksHit / (idxBlksHit + idxBlksRead)]`).
- **`lastReset`**- Der Zeitpunkt, zu dem diese Statistiken zuletzt zurückgesetzt wurden. Die von bereitgestellten Statistiken `db.collection.stats()` werden zurückgesetzt, wenn starting/stopping die Instanz geclustert oder up/down skaliert wird.

Eine Aufschlüsselung der `idxBlksRead` Felder `idxBlksHit` und für jeden Index kann auch mithilfe des `indexStats` Befehls abgerufen werden. Indexspezifische Cachestatistiken können mit dem folgenden Befehl abgerufen werden:

```
db.collection.aggregate([{$indexStats:{}}]).pretty()
```

Für jeden Index finden Sie die folgenden Cache-Statistiken unter dem `cacheStats` Feld:

- **`blksHit`**- Die Anzahl der Blöcke, die für diesen Index aus dem Cache gelesen wurden.
- **`blksRead`**- Die Anzahl der Blöcke, die für diesen Index von der Festplatte gelesen wurden.

- **`blksHitRatio`**- Die Cache-Trefferquote, auf vier Dezimalstellen gerundet, berechnet durch`100 * [blksHit / (blksHit + blksRead)]`.

Wie finde und beende ich langsame und blockierte Abfragen?

Benutzerabfragen können aufgrund eines nicht optimalen Abfrageplans langsam ausgeführt oder aufgrund von Ressourcenkonflikten blockiert werden.

Zum Suchen nach Abfragen, die aufgrund eines nicht optimalen Abfrageplans verlangsamt oder aufgrund von Ressourcenkonflikten blockiert werden, verwenden Sie den Befehl `currentOp`. Sie können den Befehl filtern, um die Liste der relevanten Abfragen einzugrenzen. Mit der langsamen Abfrage muss `opid` verknüpft sein, um sie beenden zu können.

Die folgende Abfrage verwendet den Befehl `currentOp`, um alle Abfragen aufzulisten, die blockiert oder länger als 10 Sekunden ausgeführt werden.

```
db.adminCommand({
    aggregate: 1,
    pipeline: [
        {$currentOp: {}},
        {$match:
            {$or: [
                {secs_running: {$gt: 10}},
                {WaitState: {$exists: true}}]}},
        {$project: {_id:0, opid: 1, secs_running: 1}}],
    cursor: {}
});
```

Als Nächstes können Sie die Abfrage einschränken, um die `opid` von länger als 10 Sekunden ausgeführten Abfragen zu finden und diese zu beenden.

So finden und beenden Sie eine Abfrage, die länger als 10 Sekunden läuft:

1. Suchen Sie die `opid` der Abfrage.

   ```
   db.adminCommand({
       aggregate: 1,
       pipeline: [
           {$currentOp: {}},
           {$match:
   ```

```
            {$or:
                [{secs_running: {$gt: 10}},
                 {WaitState: {$exists: true}}]}}],
    cursor: {}
});
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    "waitedMS" : NumberLong(0),
    "cursor" : {
        "firstBatch" : [
            {
                "opid" : 24646,
                "secs_running" : 12
            }
        ],
        "id" : NumberLong(0),
        "ns" : "admin.$cmd"
    },
    "ok" : 1
}
```

2. Beenden Sie die Abfrage mit der Operation `killOp`.

```
db.adminCommand({killOp: 1, op: 24646});
```

Wie kann ich einen Abfrageplan einsehen und eine Abfrage optimieren?

Wenn eine Abfrage langsam ausgeführt wird, erfordert die Abfrageausführung möglicherweise den vollständigen Scan der Sammlung, um die relevanten Dokumente auszuwählen. Manchmal kann die Abfrage schneller ausgeführt werden, wenn geeignete Indizes erstellt werden. Mit dem Befehl `explain` erkennen Sie dieses Szenario und können die Felder für die Indizes auswählen.

> **Note**
>
> Amazon DocumentDB emuliert die MongoDB 3.6-API auf einer speziell entwickelten Datenbank-Engine, die ein verteiltes, fehlertolerantes, selbstheilendes Speichersystem verwendet. Daher können sich die Abfragepläne und die Ausgabe von zwischen Amazon DocumentDB und MongoDB unterscheiden. `explain()` Kunden, die die Kontrolle über ihren

Abfrageplan wünschen, können den `$hint`-Operator verwenden, um die Auswahl eines bevorzugten Indexes zu erzwingen.

Führen Sie die Abfrage, die Sie verbessern möchten, unter dem Befehl `explain` wie folgt aus.

```
db.runCommand({explain: {<query document>}})
```

Im Folgenden finden Sie eine Beispieloperation.

```
db.runCommand({explain:{
    aggregate: "sample-document",
    pipeline: [{$match: {x: {$eq: 1}}}],
    cursor: {batchSize: 1}}
});
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    "queryPlanner" : {
        "plannerVersion" : 1,
        "namespace" : "db.test",
        "winningPlan" : {
            "stage" : "COLLSCAN"
        }
    },
    "serverInfo" : {
        "host" : "...",
        "port" : ...,
        "version" : "..."
    },
    "ok" : 1
}
```

Die Ausgabe oben zeigt an, dass in der Phase `$match` die gesamte Sammlung gescannt werden muss, um zu prüfen, ob das Feld `"x"` in jedem Dokument gleich 1 ist. Wenn die Sammlung zahlreiche Dokumente enthält, wird der Scan der Sammlung sehr langsam ausgeführt. Damit ist auch die Gesamtleistung der Abfrage sehr niedrig. Das Vorhandensein von `"COLLSCAN"` in der Ausgabe des Befehls `explain` zeigt daher an, dass die Abfrageleistung durch die Erstellung geeigneter Indizes verbessert werden kann.

In diesem Beispiel prüft die Abfrage, ob das Feld "x" in allen Dokumenten gleich 1 ist. Das Erstellen eines Indexes für das Feld "x" ermöglicht der Abfrage, einen vollständigen Scan der Sammlung zu vermeiden und den Index zu verwenden, um die relevanten Dokumente schneller zurückzugeben.

Nach dem Erstellen eines Indexes für das Feld "x" sieht die explain-Ausgabe wie folgt aus.

```
{
    "queryPlanner" : {
         "plannerVersion" : 1,
         "namespace" : "db.test",
         "winningPlan" : {
             "stage" : "IXSCAN",
             "indexName" : "x_1",
             "direction" : "forward"
         }
    },
    "serverInfo" : {
        "host" : "...",
        "port" : ...,
        "version" : "..."
    },
    "ok" : 1
}
```

Das Erstellen eines Indexes für das Feld "x" ermöglicht der Stufe $match die Verwendung eines Index, um die Anzahl der Dokumente zu reduzieren, in denen das Prädikat "x = 1" ausgewertet werden muss.

Bei kleinen Sammlungen kann sich der Amazon DocumentDB-Abfrageprozessor dafür entscheiden, keinen Index zu verwenden, wenn die Leistungssteigerungen vernachlässigbar sind.

Wie kann ich einen Abfrageplan in elastischen Clustern sehen?

Verwenden Sie den explain Befehl, um einen Abfrageplan in elastischen Clustern zu untersuchen. Im Folgenden finden Sie ein Beispiel für einen explain Vorgang mit einer Suchabfrage, die auf eine gemeinsame Sammlung abzielt:

```
db.runCommand(
   {
     explain: { find: "cities", filter: {"name": "Seoul"}}
   }
```

```
)
```

> **Note**
>
> Amazon DocumentDB emuliert MongoDB auf einer speziell entwickelten Datenbank-Engine. Daher `explain()` können sich die Abfragepläne und die Ausgabe von zwischen Amazon DocumentDB und MongoDB unterscheiden. Sie können den Abfrageplan mithilfe des `$hint` Operators steuern, um die Auswahl eines bevorzugten Indexes zu erzwingen.

Die Ausgabe dieser Operation könnte etwa wie folgt aussehen (JSON-Format):

```
{
  "queryPlanner" : {
    "elasticPlannerVersion" : 1,
    "winningPlan" : {
      "stage" : "SINGLE_SHARD",
      "shards" : [
        {
          "plannerVersion" : 1,
          "namespace" : "population.cities",
          "winningPlan" : {
            "stage" : "SHARD_MERGE",
            "shards" : [
              {
                "shardName" : "f2cf5cfd-fe9c-40ca-b4e5-298ca0d11111",
                "plannerVersion" : 1,
                "namespace" : "population.cities",
                "winningPlan" : {
                  "stage" : "PARTITION_MERGE",
                  "inputStages" : [
                    {
                      "stage" : "COLLSCAN",
                      "partitionCount" : 21
                    }
                  ]
                }
              },
              {
                "shardName" : "8f3f80e2-f96c-446e-8e9d-aab8c7f22222",
                "plannerVersion" : 1,
                "namespace" : "population.cities",
```

```
                "winningPlan" : {
                  "stage" : "PARTITION_MERGE",
                  "inputStages" : [
                    {
                      "stage" : "COLLSCAN",
                      "partitionCount" : 21
                    }
                  ]
                }
              },
              {
                "shardName" : "32c5a06f-1b2b-4af1-8849-d7c4a033333",
                "plannerVersion" : 1,
                "namespace" : "population.cities",
                "winningPlan" : {
                  "stage" : "PARTITION_MERGE",
                  "inputStages" : [
                    {
                      "stage" : "COLLSCAN",
                      "partitionCount" : 22
                    }
                  ]
                }
              }
            ]
          },
          "shardName" : "32c5a06f-1b2b-4af1-8849-d7c4a0f3fb58"
        }
      ]
    }
  },
  "serverInfo" : {
    "host" : "example-4788267630.us-east-1.docdb-elastic.amazonaws.com:27017",
    "version" : "5.0.0"
  },
  "ok" : 1,
  "operationTime" : Timestamp(1695097923, 1)
}
```

Die vorherige Ausgabe zeigt den Abfrageplan für die `find` Abfrage in einem Cluster mit drei Shards. Jeder Shard hat mehrere Datenpartitionen, die unterschiedliche Eingabeschritte haben können. In diesem Beispiel wird ein „COLLSCAN“ (ein Sammlungsscan) auf allen Partitionen ausgeführt, bevor die Ergebnisse in der“ PARTITION_MERGE „-Stufe innerhalb jedes Shards

zusammengeführt werden. Die Ergebnisse der Shards werden dann in der“ SHARD_MERGE „-Phase zusammengeführt, bevor sie an den Client zurückgesendet werden.

Wie liste ich alle laufenden Operationen für eine Instance auf?

Als Benutzer oder Hauptbenutzer möchten Sie häufig alle aktuellen Vorgänge, die auf einer Instance ausgeführt werden, zu Diagnose- und Fehlerbehebungszwecken auflisten. (Weitere Informationen zum Verwalten von Benutzern finden Sie unter Verwaltung von Amazon DocumentDB-Benutzern.)

Mit der mongo Shell können Sie die folgende Abfrage verwenden, um alle laufenden Operationen auf einer Amazon DocumentDB-Instance aufzulisten.

```
db.adminCommand({currentOp: 1, $all: 1});
```

Die Abfrage gibt die vollständige Liste aller Benutzerabfragen und internen Systemaufgaben zurück, die zurzeit auf der Instance ausgeführt werden.

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    "inprog" : [
        {
            "desc" : "INTERNAL"
        },
        {
            "desc" : "TTLMonitor",
            "active" : false
        },
        {
            "client" : ...,
            "desc" : "Conn",
            "active" : true,
            "killPending" : false,
            "opid" : 195,
            "ns" : "admin.$cmd",
            "command" : {
                "currentOp" : 1,
                "$all" : 1
            },
            "op" : "command",
            "$db" : "admin",
            "secs_running" : 0,
```

```
            "microsecs_running" : NumberLong(68),
            "clientMetaData" : {
            "application" : {
                "name" : "MongoDB Shell"
            },
            "driver" : {
                ...
            },
            "os" : {
                ...
            }
          }
       },
       {
          "desc": "GARBAGE_COLLECTION",
          "garbageCollection": {
             "databaseName": "testdb",
             "collectionName": "testCollectionA"
          },
          "secs_running": 3,
          "microsecs_running": NumberLong(3123456)
       },
       {
          "desc": "GARBAGE_COLLECTION",
          "garbageCollection": {
             "databaseName": "testdb",
             "collectionName": "testCollectionB"
          },
          "secs_running": 4,
          "microsecs_running": NumberLong(4123456)
       }
    ],
    "ok" : 1
}
```

Die folgenden Werte sind gültige Werte für das Feld `"desc"`:

- **`INTERNAL`**— Interne Systemaufgaben wie die Cursorbereinigung oder veraltete Benutzerbereinigungsaufgaben.
- **`TTLMonitor`**— Der Time to Live (TTL) -Monitor-Thread. Der Ausführungsstatus wird im Feld `"active"` dargestellt.
- **`GARBAGE_COLLECTION`**— Der interne Garbage-Collector-Thread.

- **CONN**— Die Benutzerabfrage.
- **CURSOR**— Bei der Operation handelt es sich um einen Cursor im Leerlauf, der darauf wartet, dass der Benutzer den Befehl „GetMore“ aufruft, um den nächsten Ergebnisstapel abzurufen. In diesem Zustand verbraucht der Cursor Speicher, verbraucht aber keine Rechenleistung.

In der obigen Ausgabe werden auch alle Benutzerabfragen im System aufgeführt. Jede Benutzerabfrage wird im Kontext einer Datenbank und einer Sammlung ausgeführt. Die Kombination dieser beiden Komponenten wird als Namespace bezeichnet. Der Namespace jeder Benutzerabfrage ist im Feld `"ns"` verfügbar.

Manchmal müssen Sie alle Benutzerabfragen auflisten, die in einem bestimmten Namespace ausgeführt werden. Daher muss die vorherige Ausgabe anhand des Felds `"ns"` gefiltert werden. Im Folgenden finden Sie eine Beispielabfrage für eine gefilterte Ausgabe. Die Abfrage listet alle Benutzerabfragen auf, die zurzeit in der Datenbank `"db"` und in der Sammlung `"test"` ausgeführt werden (d. h. im Namespace `"db.test"`).

```
db.adminCommand({aggregate: 1,
    pipeline: [{$currentOp: {allUsers: true, idleConnections: true}},
               {$match: {ns: {$eq: "db.test"}}}],
    cursor: {}
});
```

Als Hauptbenutzer des Systems können Sie Abfragen aller Benutzer und auch alle internen Systemaufgaben sehen. Alle anderen Benutzer können nur ihre jeweiligen Abfragen anzeigen.

Wenn die Gesamtzahl der Abfragen und internen Systemaufgaben die Standard-Batch-Cursor-Größe überschreitet, generiert die mongo-Shell automatisch das Iterator-Objekt `'it'`, um den Rest der Ergebnisse anzuzeigen. Führen Sie den Befehl `'it'` so lange aus, bis alle Ergebnisse angezeigt wurden.

Woher weiß ich, wann eine Anfrage Fortschritte macht?

Benutzerabfragen können aufgrund eines nicht optimalen Abfrageplans langsam ausgeführt oder aufgrund von Ressourcenkonflikten blockiert werden. Das Debuggen solcher Abfragen ist ein mehrstufiger Prozess, in dem möglicherweise mehrmals dieselben Schritte ausgeführt werden müssen.

Im ersten Debugging-Schritt werden alle langsamen oder blockierten Abfragen aufgeführt. Die folgende Abfrage listet alle Benutzerabfragen auf, die länger als 10 Sekunden ausgeführt wurden oder auf Ressourcen warten.

```
db.adminCommand({aggregate: 1,
                 pipeline: [{$currentOp: {}},
                            {$match: {$or: [{secs_running: {$gt: 10}},
                                            {WaitState: {$exists: true}}]}},
                            {$project: {_id:0,
                                        opid: 1,
                                        secs_running: 1,
                                        WaitState: 1,
                                        blockedOn: 1,
                                        command: 1}}],
                 cursor: {}
                });
```

Wiederholen Sie die vorherige Abfrage regelmäßig, um zu ermitteln, ob sich die Liste der Abfragen ändert, und die langsamen oder blockierten Abfragen zu identifizieren.

Wenn das Ausgabedokument der betreffenden Abfrage das Feld `WaitState` enthält, zeigt dies an, dass die Abfrage aufgrund von Ressourcenkonflikten langsam ist oder blockiert wird. Der Ressourcenkonflikt kann entweder auf I/O interne Systemaufgaben oder andere Benutzerabfragen zurückzuführen sein.

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    "waitedMS" : NumberLong(0),
    "cursor" : {
        "firstBatch" : [
            {
                "opid" : 201,
                "command" : {
                    "aggregate" : ...
                },
                "secs_running" : 208,
                "WaitState" : "IO"
            }
        ],
        "id" : NumberLong(0),
        "ns" : "admin.$cmd"
```

```
    },
    "ok" : 1
}
```

I/O kann ein Engpass sein, wenn viele Abfragen in verschiedenen Sammlungen gleichzeitig auf derselben Instanz ausgeführt werden oder wenn die Instanzgröße für den Datensatz, auf dem die Abfrage ausgeführt wird, zu klein ist. Wenn es sich bei den Abfragen um schreibgeschützte Abfragen handelt, können Sie der eben beschriebenen Situation durch Aufteilung der Abfragen für die einzelnen Sammlungen auf separate Replikate abhelfen. Wenn in verschiedenen Sammlungen gleichzeitig Updates ausgeführt werden oder die Instance zu klein für den Datensatz ist, können Sie die Instance aufwärts skalieren.

Wenn der Ressourcenkonflikt auf Abfragen anderer Benutzer zurückzuführen ist, zeigt das Feld `"blockedOn"` im Ausgabedokument die `"opid"` der Abfrage an, die sich auf die Abfrage auswirkt. Folgen Sie mithilfe der `"opid"` der Kette der Felder `"WaitState"` und `"blockedOn"` aller Abfragen, um die Abfrage an der Ende der Kette zu finden.

Wenn es sich bei der Aufgabe am Ende der Kette um eine interne Aufgabe handelt, kann das Problem nur behoben werden, indem die Abfrage beendet und später erneut ausgeführt wird.

Im Folgenden finden Sie eine Beispielausgabe, in der die Suchabfrage auf einer Sammlungssperre blockiert wird, die im Besitz einer anderen Aufgabe ist.

```
{
    "inprog" : [
        {
            "client" : "...",
            "desc" : "Conn",
            "active" : true,
            "killPending" : false,
            "opid" : 75,
            "ns" : "...",
            "command" : {
                "find" : "...",
                "filter" : {

                }
            },
            "op" : "query",
            "$db" : "test",
            "secs_running" : 9,
            "microsecs_running" : NumberLong(9449440),
```

```
            "threadId" : 24773,
            "clientMetaData" : {
                "application" : {
                   "name" : "MongoDB Shell"
                },
                "driver" : {
                    ...
                },
                "os" : {
                    ...
                }
            },
            "WaitState" : "CollectionLock",
            "blockedOn" : "INTERNAL"
        },
        {
            "desc" : "INTERNAL"
        },
        {
            "client" : "...",
            ...
            "command" : {
                "currentOp" : 1
            },
            ...
        }
    ],
    "ok" : 1
}
```

Wenn `"WaitState"` die Werte `"Latch"`, `"SystemLock"`, `"BufferLock"`, `"BackgroundActivity"` oder `"Other"` hat, wird der Ressourcenkonflikt von internen Systemaufgaben verursacht. Wenn die Situation lange andauert, besteht die einzige Abhilfe darin, die Abfrage zu beenden und später erneut auszuführen.

Wie stelle ich fest, warum ein System plötzlich langsam läuft?

Im Folgenden finden Sie einige häufige Gründe für die Verlangsamung von Systemen:

- Übermäßige Ressourcenkonflikte zwischen gleichzeitigen Abfragen
- Zunahme der Anzahl der aktiven gleichzeitigen Abfragen im Laufe der Zeit
- Interne Systemaufgaben wie `"GARBAGE_COLLECTION"`

Wenn Sie die Systemnutzung im Laufe der Zeit überwachen möchten, führen Sie in regelmäßigen Abständen die folgende `"currentOp"`-Abfrage aus, und geben Sie die Ergebnisse in einem externen Speicher aus. Die Abfrage zählt die Anzahl der Abfragen und Operationen in jedem Namespace im System. Sie können anschließend die Systemnutzungsergebnisse analysieren, um die Systemlast zu verstehen und entsprechende Maßnahmen zu ergreifen.

```
db.adminCommand({aggregate: 1,
                 pipeline: [{$currentOp: {allUsers: true, idleConnections: true}},
                            {$group: {_id: {desc: "$desc", ns: "$ns", WaitState:
 "$WaitState"}, count: {$sum: 1}}}],
                 cursor: {}
                });
```

Diese Abfrage gibt eine Aggregation aller in den einzelnen Namespaces ausgeführten Abfragen, alle internen Systemaufgaben und die eindeutige Anzahl der Wartezustände (wenn vorhanden) pro Namespace zurück.

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{
    "waitedMS" : NumberLong(0),
    "cursor" : {
        "firstBatch" : [
            {
                "_id" : {
                    "desc" : "Conn",
                    "ns" : "db.test",
                    "WaitState" : "CollectionLock"
                },
               "count" : 2
            },
            {
                "_id" : {
                    "desc" : "Conn",
                    "ns" : "admin.$cmd"
                },
                "count" : 1
            },
            {
                "_id" : {
                    "desc" : "TTLMonitor"
                },
```

```
                "count" : 1
            }
        ],
        "id" : NumberLong(0),
        "ns" : "admin.$cmd"
    },
    "ok" : 1
}
```

In der oben gezeigten Ausgabe werden zwei Benutzerabfragen im Namespace `"db.test"` durch eine Sammlungssperre blockiert: 1 Abfrage in Namespace `"admin.$cmd"` und eine interne `"TTLMonitor"`-Aufgabe.

Wenn die Ausgabe viele Abfragen mit blockierenden Wartezuständen anzeigt, finden Sie Informationen unter Wie finde und beende ich langsame und blockierte Abfragen?.

Wie ermittle ich die Ursache für eine hohe CPU-Auslastung auf einer oder mehreren Cluster-Instances?

In den folgenden Abschnitten finden Sie Informationen, die Ihnen möglicherweise helfen, die Ursache für die hohe CPU-Nutzung einer Instance zu ermitteln. Die Ergebnisse sind vom Workload abhängig.

- Informationen dazu, wie Sie ermitteln, warum eine Instance plötzlich langsam ausgeführt wird, finden Sie unter Wie stelle ich fest, warum ein System plötzlich langsam läuft?.
- Informationen zum Identifizieren und Beenden langsamer Abfragen auf einer bestimmten Instance finden Sie unter Wie finde und beende ich langsame und blockierte Abfragen?.
- Informationen dazu, wie Sie ermitteln, ob eine Abfrage fortschreitet, finden Sie unter Woher weiß ich, wann eine Anfrage Fortschritte macht?.
- Informationen dazu, warum die Ausführung einer Abfrage lange dauert, finden Sie unter Wie kann ich einen Abfrageplan einsehen und eine Abfrage optimieren?.
- Informationen dazu, wie Sie langsame Abfragen über die Zeit nachverfolgen, finden Sie unter Profilierung von Amazon DocumentDB-Vorgängen.

Abhängig vom Grund für die hohe CPU-Nutzung Ihrer Instance können Sie einen oder mehrere der folgenden Schritte ausführen.

- Wenn die primäre Instance eine hohe CPU-Nutzung aufweist, die Replikat-Instances jedoch nicht, sollten Sie den Datenverkehr für Lesevorgänge über die Client-Leseeinstellungen auf

mehrere Replikate verteilen (z. B. `secondaryPreferred`). Weitere Informationen finden Sie unter Verbindung zu Amazon DocumentDB als Replikatsatz herstellen.

Die Verwendung von Replikaten für Lesevorgänge kann zu einer besseren Nutzung der Cluster-Ressourcen führen, da die primäre Instance eine größere Menge an Schreibdatenverkehr verarbeiten kann. Lesevorgänge aus -Replikaten sind Eventually Consistent.

- Wenn die hohe CPU-Nutzung durch Schreib-Workloads verursacht wird, können Sie durch die Skalierung der Cluster-Instances auf einen größeren Instance-Typ die Anzahl der CPU-Kerne erhöhen, die für den Workload verfügbar sind. Weitere Informationen erhalten Sie unter Verwaltung von Amazon DocumentDB DocumentDB-Instances und Spezifikationen der Instanzklasse.
- Wenn alle Cluster-Instances eine hohe CPU-Nutzung aufweisen und der Workload Replikate für Lesevorgänge verwendet, erhöht das Hinzufügen weiterer Replikate zum Cluster die Zahl der für Lesevorgänge verfügbaren Ressourcen. Weitere Informationen finden Sie unter Hinzufügen einer Amazon DocumentDB DocumentDB-Instance zu einem Cluster.

Wie ermittle ich die offenen Cursor auf einer Instance?

Wenn Sie mit einer Amazon DocumentDB-Instance verbunden sind, können Sie den Befehl verwenden, `db.runCommand("listCursors")` um die geöffneten Cursor auf dieser Instance aufzulisten. In einer bestimmten Amazon DocumentDB-Instance gibt es je nach Instance-Typ ein Kontingent von bis zu 4.560 aktiven Cursorn, die zu einem bestimmten Zeitpunkt geöffnet sind. Schließen Sie Cursor, die nicht mehr verwendet werden, da Cursor Ressourcen auf einer Instance nutzen. Weitere Informationen finden Sie unter Amazon DocumentDB-Kontingente.

```
db.runCommand("listCursors")
```

Wie ermittle ich die aktuelle Version der Amazon DocumentDB-Engine?

Führen Sie den folgenden Befehl aus, um Ihre aktuelle Amazon DocumentDB-Engine-Version zu ermitteln.

```
db.runCommand({getEngineVersion: 1})
```

Die Ausgabe dieser Operation sieht in etwa folgendermaßen aus (JSON-Format).

```
{ "engineVersion" : "2.x.x", "ok" : 1 }
```

> **Note**
>
> Die Engine-Version für Amazon DocumentDB 3.6 ist 1.x.x, die Engine-Version für Amazon DocumentDB 4.0 ist 2.x.x, die Engine-Version für Amazon DocumentDB 5.0 ist 3.x.x und die Engine-Version für Amazon DocumentDB 8.0 ist 4.x.x.

Wie analysiere ich die Indexnutzung und identifiziere ungenutzte Indizes?

Zum Identifizieren der Indizes für eine bestimmte Sammlung führen Sie den folgenden Befehl aus:

```
db.collection.getIndexes()
```

Um zu analysieren, wie viele Indizes bei Operationen an den Sammlungen verwendet werden, können die `indexStats` Befehle `collStats` und verwendet werden. Führen Sie den folgenden Befehl aus, um die Gesamtzahl der mithilfe von Indizes (Indexscans) durchgeführten Scans im Vergleich zur Anzahl der ohne Index durchgeführten Scans (Sammlungsscans) anzuzeigen:

```
db.collection.stats()
```

Die Ausgabe für diesen Befehl enthält die folgenden Werte:

- **`idxScans`**— Die Anzahl der anhand eines Indexes für diese Sammlung durchgeführten Scans.
- **`collScans`**- Die Anzahl der Scans, die für diese Sammlung durchgeführt wurden, ohne einen Index zu verwenden. Bei diesen Scans hätten die Dokumente in der Sammlung nacheinander durchsucht werden müssen.
- **`lastReset`**- Der Zeitpunkt, zu dem diese Zähler das letzte Mal zurückgesetzt wurden. Die durch diesen Befehl bereitgestellten Statistiken werden zurückgesetzt, wenn starting/stopping die Instanz geclustert oder up/down skaliert wird.

Eine Aufschlüsselung darüber, wie stark jeder Index verwendet wird, finden Sie in der Ausgabe des folgenden Befehls. Es hat sich bewährt, ungenutzte Indizes regelmäßig zu identifizieren und zu entfernen, um die Leistung zu verbessern und die Kosten zu senken, da auf diese Weise unnötige Rechenleistung, Speicherung und I/Os Verwaltung der Indizes vermieden werden.

```
db.collection.aggregate([{$indexStats:{}}]).pretty()
```

Die Ausgabe dieses Befehls gibt die folgenden Werte für jeden Index an, der für die Sammlung erstellt wurde:

- **`ops`**— Die Anzahl der Operationen, die den Index verwendet haben. Wenn Ihre Workload über einen ausreichend langen Zeitraum ausgeführt wurde und Sie sicher sind, dass sich Ihre Workload in einem konstanten Zustand befindet, würde ein `ops`-Wert von Null anzeigen, dass der Index überhaupt nicht verwendet wird.
- **`numDocsRead`**- Die Anzahl der Dokumente, die bei Vorgängen mit diesem Index gelesen wurden.
- **`since`**— Die Zeit, seit Amazon DocumentDB mit der Erfassung von Statistiken zur Indexnutzung begonnen hat. Dies ist in der Regel der Wert seit dem letzten Datenbankneustart oder der letzten Wartungsaktion.
- **`size`**- Die Größe dieses Indexes in Byte.

Das folgende Beispiel ist eine Beispielausgabe bei der Ausführung des obigen Befehls:

```
{
    "name" : "_id_",
    "key" : {
        "_id" : 1
    },
    "host" : "example-host.com:12345",
    "size" : NumberLong(...),
    "accesses" : {
        "ops" : NumberLong(...),
        "docsRead" : NumberLong(...),
        "since" : ISODate("...")
    },
    "cacheStats" : {
        "blksRead" : NumberLong(...),
        "blksHit" : NumberLong(...),
        "hitRatio" : ...
    }
}
{
    "name" : "x_1",
    "key" : {
        "x" : 1
    },
    "host" : "example-host.com:12345",
    "size" : NumberLong(...),
```

```
    "accesses" : {
        "ops" : NumberLong(...),
        "docsRead" : NumberLong(...),
        "since" : ISODate("...")
    },
    "cacheStats" : {
        "blksRead" : NumberLong(...),
        "blksHit" : NumberLong(...),
        "hitRatio" : ...
    }
}
```

Um die Gesamtindexgröße für eine Sammlung zu ermitteln, führen Sie den folgenden Befehl aus:

```
db.collection.stats()
```

Führen Sie den folgenden Befehl aus, um einen nicht verwendeten Index zu löschen:

```
db.collection.dropIndex("indexName")
```

Wie identifiziere ich fehlende Indizes?

Sie können den Amazon DocumentDB-Profiler verwenden, um langsame Abfragen zu protokollieren. Eine Abfrage, die wiederholt im Protokoll über langsame Abfragen erscheint, kann darauf hinweisen, dass ein zusätzlicher Index erforderlich ist, um diese Abfrageleistung zu verbessern.

Sie können Möglichkeiten für hilfreiche Indizes identifizieren, indem Sie nach Abfragen mit langer Laufzeit suchen, die eine oder mehrere Stufen enthalten, die mindestens eine `COLLSCAN` Phase ausführen. Das bedeutet, dass die Abfragephase jedes Dokument in der Sammlung lesen muss, um eine Antwort auf die Abfrage zu geben.

Das folgende Beispiel zeigt eine Abfrage zu einer Sammlung von Taxifahrten, die auf einer großen Sammlung ausgeführt wurden.

```
db.rides.count({"fare.totalAmount":{$gt:10.0}}))
```

Um dieses Beispiel auszuführen, musste die Abfrage einen Sammlungsscan durchführen (d. h. jedes einzelne Dokument in der Sammlung lesen), da es keinen Index für das `fare.totalAmount`-Feld gibt. Die Ausgabe des Amazon DocumentDB-Profilers für diese Abfrage sieht in etwa wie folgt aus:

```
{
    ...
    "cursorExhausted": true,
    "nreturned": 0,
    "responseLength": 0,
    "protocol": "op_query",
    "millis": 300679,
    "planSummary": "COLLSCAN",
    "execStats": {
        "stage": "COLLSCAN",
        "nReturned": "0",
        "executionTimeMillisEstimate": "300678.042"
    },
    "client": "172.31.5.63:53878",
    "appName": "MongoDB Shell",
    "user": "example"
}
```

Um die Abfrage in diesem Beispiel zu beschleunigen, möchten Sie einen Index auf `fare.totalAmount` erstellen, wie unten gezeigt.

```
db.rides.createIndex( {"fare.totalAmount": 1}, {background: true} )
```

> **Note**
>
> Indizes, die im Vordergrund erstellt werden (d. h. wenn die Option `{background:true}` beim Erstellen des Index nicht angegeben wurde), erhalten eine exklusive Schreibsperre, die Anwendungen daran hindert, Daten so lange in die Sammlung zu schreiben, bis der Indexaufbau abgeschlossen ist. Beachten Sie diese mögliche Auswirkung beim Erstellen von Indizes für Produktions-Cluster. Wenn Sie Indizes erstellen, legen Sie fest. `{background:true}`

Im Allgemeinen möchten Sie Indizes für Felder mit hoher Kardinalität erstellen (z. B. eine große Anzahl eindeutiger Werte). Das Erstellen eines Index für ein Feld mit geringer Kardinalität kann zu einem großen Index führen, der nicht verwendet wird. Der Amazon DocumentDB-Abfrageoptimierer berücksichtigt bei der Erstellung eines Abfrageplans die Gesamtgröße der Sammlung und die Selektivität der Indizes. Manchmal wird der Abfrageprozessor einen `COLLSCAN` auswählen, selbst wenn ein Index vorhanden ist. Dies geschieht, wenn der Abfrageprozessor schätzt, dass die

Verwendung des Index keinen Leistungsvorteil gegenüber dem Scannen der gesamten Sammlung bringt. Wenn Sie den Abfrageprozessor zwingen möchten, einen bestimmten Index zu verwenden, können Sie den `hint()`-Operator wie unten gezeigt verwenden.

```
db.collection.find().hint("indexName")
```

Wie stelle ich fest, dass die Datenbanksammlung überfüllt ist?

Eine Aufblähung der Sammlung tritt auf, wenn die Größe einer Sammlung aufgrund der Anhäufung toter oder veralteter Dokumente oder der Fragmentierung innerhalb von Datenbankseiten zunimmt. Der angegebene Prozentsatz gibt den Speicherplatz an Dokumenten an, der für zukünftige Dokumente genutzt werden kann. Diese Überlastung verbraucht Speicherplatz sowohl im Puffercache als auch im Speicher. Um die Überlastung zu vermeiden, müssen Sammlungen entweder über dump/restore oder mithilfe eines Migrations-Loopbacks und Switchovers während eines Wartungsfensters neu geladen werden.

Example Beispiel

Führen Sie den folgenden Befehl aus, um den ungenutzten Speicherplatz für Ihre Sammlung zu ermitteln:

```
db.runCommand({collStats:'coll'})
```

Das Ergebnis sieht ungefähr so aus:

```
{
        "ns" : "test.coll",
        "count" : 7500,
        "size" : 23250,
        "avgObjSize" : 31,
        "storageSize" : 106496,
        "unusedStorageSize" : {
                "unusedBytes" : 16384,
                "unusedPercent" : 25.12
        },
        "compression" : {
                "enable" : false
        },
        "capped" : false,
        "nindexes" : 1,
```

```
        "totalIndexSize" : 57344,
        "indexSizes" : {
                "_id_" : 57344
        },
        "collScans" : 4,
        "idxScans" : 10000,
        "opCounter" : {
                "numDocsIns" : 1000,
                "numDocsUpd" : 0,
                "numDocsDel" : 250
        },
        "cacheStats" : {
                "collBlksHit" : 3570,
                "collBlksRead" : 8,
                "collHitRatio" : 99.7765,
                "idxBlksHit" : 12293,
                "idxBlksRead" : 6,
                "idxHitRatio" : 99.9513
        },
        "lastReset" : "2024-12-18 00:30:21.552019+00",
        "ok" : 1,
        "operationTime" : Timestamp(1734632375, 1)
}
```

Zusammenfassung nützlicher Abfragen

Die folgenden Abfragen können für die Überwachung der Leistung und der Ressourcenauslastung in Amazon DocumentDB hilfreich sein.

- Verwenden Sie den folgenden Befehl, um Statistiken zu einer bestimmten Sammlung anzuzeigen, einschließlich Betriebszählern, Cache-Statistiken, Zugriffsstatistiken und Größenstatistiken:

  ```
  db.collection.stats()
  ```

- Verwenden Sie den folgenden Befehl, um Statistiken zu jedem Index anzuzeigen, der für eine Sammlung erstellt wurde, einschließlich der Größe des Indexes, indexspezifischer Cachestatistiken und Statistiken zur Indexnutzung:

  ```
  db.collection.aggregate([{$indexStats:{}}]).pretty()
  ```

- Verwenden Sie die folgende Abfrage, um alle Aktivitäten aufzulisten.

```
db.adminCommand({currentOp: 1, $all: 1});
```

- Der folgende Code listet alle langsamen oder blockierten Abfragen auf.

```
db.adminCommand({aggregate: 1,
                 pipeline: [{$currentOp: {}},
                            {$match: {$or: [{secs_running: {$gt: 10}},
                                            {WaitState: {$exists: true}}]}},
                            {$project: {_id:0,
                                        opid: 1,
                                        secs_running: 1,
                                        WaitState: 1,
                                        blockedOn: 1,
                                        command: 1}}],
                 cursor: {}
                });
```

- Der folgende Code beendet eine Abfrage.

```
db.adminCommand({killOp: 1, op: <opid of running or blocked query>});
```

- Verwenden Sie den folgenden Code, um eine aggregierte Ansicht des Systemstatus zu erhalten.

```
db.adminCommand({aggregate: 1,
                 pipeline: [{$currentOp: {allUsers: true, idleConnections: true}},
                            {$group: {_id: {desc: "$desc", ns: "$ns", WaitState:
 "$WaitState"}, count: {$sum: 1}}}],
                 cursor: {}
                });
```

Müllabfuhr in Amazon DocumentDB

Amazon DocumentDB implementiert eine MVCC-Datenbankarchitektur (Multiversion Concurrency Control), die für jeden Aktualisierungsvorgang neue Versionen von Dokument- und Indexeinträgen erstellt. Diese Architektur ermöglicht die Isolierung von Transaktionen und verhindert, dass die Änderungen einer Transaktion in einer anderen erscheinen.

Topics

- Grundlegendes zur Speicherbereinigung in Amazon DocumentDB

- Prozess der Müllabfuhr
- Speicherarchitektur und erweiterter Speicher
- Überwachung der Müllabfuhr
- Beispiel für eine CollStats-Ausgabe
- Häufig gestellte Fragen

Grundlegendes zur Speicherbereinigung in Amazon DocumentDB

Die Garbage Collection (GC) ist ein automatisierter Hintergrundprozess, der die optimale Systemleistung und Verfügbarkeit in Amazon DocumentDB gewährleistet. Wie viele moderne Datenbanken erstellt auch die MVCC-Architektur von Amazon DocumentDB bei jedem Update neue Dokument- und Indexversionen. Jeder Schreibvorgang benötigt eine eindeutige MVCC-ID von einem endlichen Zähler. Diese IDs identifizieren, zu welcher Transaktion eine Dokumentversion gehört und ob sie festgeschrieben oder zurückgesetzt wurde. Im Laufe der Zeit häufen sich diese alten Versionen und ihre MVCC-IDs an und müssen bereinigt werden, um Leistungseinbußen zu verhindern.

Funktionen der Garbage-Collection

Der Garbage Collector erfüllt drei wesentliche Funktionen:

- Gewinnt Speicherplatz zurück — Er entfernt veraltete Dokument- und Indexversionen, die für aktive Abfragen nicht mehr benötigt werden, und gibt so Speicherplatz für zukünftige Schreibvorgänge frei.
- Beugt einem MVCC-ID-Überlauf vor — Es verhindert einen MVCC-ID-Überlauf, indem der endliche Zähler von MVCC-IDs verwaltet wird. Ohne diese Verwaltung würde der Zähler irgendwann sein Limit erreichen, sodass die Datenbank in einen temporären Lesemodus versetzt wird, bis die IDs wiederverwendet werden.
- Beibehaltung der Abfrageleistung — Es sorgt für eine optimale Abfrageleistung, indem tote Dokumentversionen entfernt werden, die sich sonst ansammeln und die Abfrageverarbeitung verlangsamen würden.

Prozess der Müllabfuhr

Der GC-Prozess wird pro Sammlung ausgeführt und kann mehrere Prozesse gleichzeitig in verschiedenen Sammlungen ausführen. Der Prozess besteht aus vier aufeinanderfolgenden Phasen:

1. Identifizierung — Das System identifiziert Dokument- und Indexversionen, auf die durch aktive Transaktionen oder Abfragen nicht mehr verwiesen wird.
2. Laden des Speichers — Alte Dokumente und Indexeinträge werden in den Speicher geladen, sofern sie nicht bereits vorhanden sind.
3. Löschen — Veraltete Versionen werden dauerhaft gelöscht, um Speicherplatz freizugeben.
4. MVCC-ID-Recycling — Das System recycelt MVCC-IDs aus gelöschten Versionen für neue Operationen.

Wenn die Garbage Collection die Verarbeitung alter Dokumentversionen abgeschlossen hat, werden die ältesten MVCC-IDs aus dem System entfernt. Diese Bereinigung ist entscheidend, um einen Überlauf der MVCC-IDs zu verhindern, indem MVCC-IDs recycelt und für neue Schreibvorgänge im gesamten Cluster verfügbar gemacht werden. Ohne diesen Recyclingprozess würde das System irgendwann seinen begrenzten MVCC-ID-Zähler erschöpfen und in einen schreibgeschützten Zustand übergehen.

Planung der Müllabfuhr

Die Müllabfuhr läuft in regelmäßigen Abständen automatisch im Hintergrund. Der Zeitpunkt und die Häufigkeit passen sich dynamisch an die Systemlast, die verfügbaren Ressourcen, das Schreibvolumen und den Verbrauch der MVCC-ID an. Bei hoher Schreibaktivität wird der GC-Prozess häufiger ausgeführt, um die gestiegene Anzahl von Dokumentversionen zu bewältigen.

Speicherarchitektur und erweiterter Speicher

Amazon DocumentDB verwendet eine ausgeklügelte Speicherarchitektur, die den Dokumentenspeicher in zwei verschiedene Segmente unterteilt:

Basis-Speichersegment

Das Basisspeichersegment enthält die primären Dokumentdaten und Metadaten. In diesem Segment werden gespeichert:

- Dokumentinhalt, der in die Standardseitengröße (8 KB) passt.
- Metadaten und Strukturinformationen des Dokuments.
- Primärindizes und ihre Einträge.
- Collection-level Statistik und Konfiguration.

Erweitertes Speichersegment

Das erweiterte Speichersegment verwendet einen speziellen Objektspeicher für große Dokumente, der für die Verarbeitung von Dokumenten konzipiert ist, die die Standardspeicherseitengröße überschreiten. Dieses Segment bietet:

- Effizienter Umgang mit großen Dokumenten — Dokumente, die den Basisspeicherschwellenwert überschreiten, werden automatisch in das erweiterte Speichersegment verschoben.
- Optimiertes Speicherlayout — Das Segment verwendet ein anderes Speicherformat, das für große Objekte optimiert ist. Dadurch wird die Fragmentierung reduziert und die Zugriffsmuster verbessert.
- Unabhängige Speicherbereinigung — Das erweiterte Speichersegment verfügt über einen eigenen Garbage-Collection-Prozess, der unabhängig von der grundlegenden Speicherbereinigung ausgeführt werden kann.
- Transparenter Zugriff — Anwendungen greifen nahtlos auf große Dokumente zu, ohne wissen zu müssen, welches Speichersegment die Daten enthält.

Das erweiterte Speichersegment ist besonders vorteilhaft für:

- Sammlungen mit Dokumenten, die große eingebettete Arrays enthalten.
- Dokumente mit umfangreichen verschachtelten Strukturen.
- Sammlungen, in denen Binärdaten oder große Textfelder gespeichert werden.
- Anwendungen mit gemischten Dokumentgrößen, bei denen einige Dokumente die Durchschnittsgröße deutlich überschreiten.

Überwachung der Müllabfuhr

Metriken auf Clusterebene

`AvailableMVCCIds`

- Standort — Amazon CloudWatch
- Beschreibung — Ein Zähler, der die Anzahl der verbleibenden Schreibvorgänge anzeigt, die ab einer Höchstgrenze von 1,8 Milliarden verfügbar sind. Wenn dieser Zähler Null erreicht, wechselt Ihr Cluster in den schreibgeschützten Modus, bis die IDs abgerufen und wiederverwendet werden. Der Zähler sinkt mit jedem Schreibvorgang und steigt, wenn die Garbage Collection alte MVCC-IDs wiederverwendet.

- Empfehlung — Stellen Sie einen Alarm ein, wenn der Wert bei Amazon DocumentDB 5.0 und höher unter 900 Millionen oder bei Amazon DocumentDB 3.6 und 4.0 unter 1,3 Milliarden fällt. Diese Frühwarnung ermöglicht es Ihnen, die empfohlenen Maßnahmen zu ergreifen, die später besprochen werden.

`LongestActiveGCRuntime`

- Standort — Amazon CloudWatch
- Beschreibung — Dauer des längsten aktiven Garbage-Collection-Vorgangs in Sekunden. Wird jede Minute aktualisiert und verfolgt nur aktive Vorgänge, ausgenommen Prozesse, die innerhalb des Zeitfensters von einer Minute abgeschlossen werden.
- Empfehlung — Vergleichen Sie mit `gcRuntimeStats` historischen Daten, um abnormales Garbage-Collection-Verhalten zu identifizieren, z. B. längere Laufzeiten bei Massenlöschungen.

Metriken auf Sammelebene

`MVCCIDStats: MVCCIdScale`

- Standort — Befehl CollStats für die Datenbank
- Beschreibung — Misst das MVCC-ID-Alter auf einer Skala von 0 bis 1, wobei 1 das Höchstalter angibt, bevor ein Cluster in den schreibgeschützten Zustand übergeht. Verwenden Sie diese Metrik zusammen`AvailableMVCCIds`, um Sammlungen zu identifizieren, die die ältesten MVCC-IDs enthalten, die den Cluster altern lassen.
- Empfehlung — Behalten Sie für jede Sammlung Werte unter 0,3 bei.

`gcRuntimeStats`

- Standort — Befehl CollStats für die Datenbank
- Beschreibung — Stellt einen zweimonatigen Verlauf der Garbage Collection-Metriken bereit, einschließlich der Gesamtzahl der Durchläufe, der durchschnittlichen Dauer und der maximalen Dauer. Schließt nur Garbage Collection-Vorgänge ein, die länger als fünf Minuten dauern, um aussagekräftige Statistiken zu gewährleisten.

`storageSizeStats`

- Standort — Befehl CollStats für die Datenbank

- Beschreibung — Bietet eine detaillierte Aufschlüsselung der Speichernutzung in den verschiedenen Speichersegmenten:
  - `storageSegmentBase`— Speicher, der vom Basisspeichersegment für Standarddokumente genutzt wird
  - `storageSegmentExtended`— Speicher, der vom erweiterten Speichersegment für große Dokumente genutzt wird
- Nutzung — Hilft bei der Identifizierung von Sammlungen mit sehr großem Dokumentenspeicher und hilft, die Speicherverteilungsmuster zu verstehen.

`unusedStorageSize`(Sammlungsebene)

- Standort — Befehl CollStats für die Datenbank
- Beschreibung — Schätzt den ungenutzten Speicherplatz in einer Sammlung auf der Grundlage von Stichprobenstatistiken. Es beinhaltet Speicherplatz aus gelöschten Dokumenten und leeren Segmenten. Die Metrik liefert sowohl kombinierte Gesamtwerte als auch Aufschlüsselungen pro Segment:
  - Kombiniert `unusedBytes` und für `unusedPercent` alle Speichersegmente
  - `storageSegmentBase`— Ungenutzter Speicherplatz speziell im Basisspeichersegment
  - `storageSegmentExtended`— Ungenutzter Speicherplatz, insbesondere im erweiterten Speichersegment

`documentFragmentStats`

- Standort — Befehl CollStats für die Datenbank
- Beschreibung — Bietet detaillierte Informationen zu Dokumentfragmenten und toten Daten in Sammlungen. Dokumentfragmente stellen die internen Speichereinheiten dar, die von der Datenbank-Engine verwendet werden, und tote Fragmente weisen auf Daten hin, auf die nicht mehr zugegriffen werden kann, die aber noch nicht abgerufen wurden. Diese Metrik beinhaltet:
  - `totalDocFragmentsCount`— Gesamtzahl der Dokumentfragmente in der Sammlung
  - `deadDocFragmentsCount`— Anzahl der Fragmente, die tote (unzugängliche) Daten enthalten
  - `deadDocFragmentsPercent`— Prozentsatz der Fragmente, die tote Daten enthalten
  - `deadDocFragmentBytes`— Geschätzter Byteverbrauch durch tote Dokumentfragmente
  - Per-segment Aufschlüsselung für `storageSegmentBase` und `storageSegmentExtended`

- Nutzung — Überwachen Sie diese Kennzahl, um zu verstehen, wie effektiv die Müllabfuhr ist, und um zu ermitteln, welche Sammlungen von Wartungsarbeiten profitieren könnten. Ein hoher Prozentsatz toter Fragmente deutet darauf hin, dass die Müllabfuhr möglicherweise ins Hintertreffen gerät oder dass die Müllabfuhr von einer Optimierung profitieren würde.

Kennzahlen auf Indexebene

`unusedStorageSize`(Indexebene)

- Standort — Befehl „IndexStats“ für die Datenbank
- Beschreibung — Schätzt den ungenutzten Speicherplatz in einem Index auf der Grundlage von Stichprobenstatistiken. Es beinhaltet Speicherplatz aus veralteten Indexeinträgen und leeren Segmenten.
- Empfehlung — Verwenden Sie den `reIndex` Befehl, um Indizes ohne Ausfallzeiten neu zu erstellen und ungenutzten Speicherplatz zurückzugewinnen. Weitere Informationen finden Sie unter Indizes verwalten.

Beispiel für eine CollStats-Ausgabe

Das folgende Beispiel zeigt eine typische `collStats` Ausgabe mit Garbage-Collection- und Speichermetriken:

```
{
    "ns" : "Mvcc_consumption_test_db.mvcc_test_collection",
    "MVCCIdStats" : {
        "MVCCIdScale" : 0.03
    },
    "gcRuntimeStats" : {
        "numRuns" : 1,
        "historicalAvgRuntime" : 3295,
        "historicalMaxRuntime" : 3295,
        "lastRuntime" : 3295,
        "lastRuntimeStart" : ISODate("2025-06-24T08:47:14Z")
    },
    "documentFragmentStats" : {
        "totalDocFragmentsCount" : 45000000,
        "deadDocFragmentsCount" : 2250000,
        "deadDocFragmentsPercent" : 5.0,
        "deadDocFragmentBytes" : 98304000,
```

```
        "storageSegmentBase" : {
            "totalDocFragmentsCount" : 30000000,
            "deadDocFragmentsCount" : 1500000,
            "deadDocFragmentsPercent" : 5.0,
            "deadDocFragmentBytes" : 65536000
        },
        "storageSegmentExtended" : {
            "totalDocFragmentsCount" : 15000000,
            "deadDocFragmentsCount" : 750000,
            "deadDocFragmentsPercent" : 5.0,
            "deadDocFragmentBytes" : 32768000
        }
    },
    "collScans" : 14,
    "count" : 30000000,
    "size" : 1320000000,
    "avgObjSize" : 44,
    "storageSize" : 6461497344,
    "storageSizeStats" : {
        "storageSegmentBase" : 4307664896,
        "storageSegmentExtended" : 2153832448
    },
    "capped" : false,
    "nindexes" : 2,
    "totalIndexSize" : 9649553408,
    "indexSizes" : {
        "_id_" : 1910661120,
        "c_1" : 7738892288
    },
    "unusedStorageSize" : {
        "unusedBytes" : 4201881600,
        "unusedPercent" : 65.05,
        "storageSegmentBase" : {
            "unusedBytes" : 2801254400,
            "unusedPercent" : 65.05
        },
        "storageSegmentExtended" : {
            "unusedBytes" : 1400627200,
            "unusedPercent" : 65.05
        }
    },
    "cacheStats" : {
        "collBlksHit" : 171659016,
        "collBlksRead" : 754061,
```

```
        "collHitRatio" : 99.5627,
        "idxBlksHit" : 692563636,
        "idxBlksRead" : 1177921,
        "idxHitRatio" : 99.8303
    },
    "idxScans" : 41823984,
    "opCounter" : {
        "numDocsIns" : 0,
        "numDocsUpd" : 20911992,
        "numDocsDel" : 0
    },
    "lastReset" : "2025-06-24 05:57:08.219711+00",
    "ok" : 1,
    "operationTime" : Timestamp(1750968826, 1)
}
```

Häufig gestellte Fragen

Wie erkenne ich, ob die Garbage Collection nicht effizient funktioniert?

Beachten Sie diese Warnzeichen, die auf eine ineffiziente Müllabfuhr hinweisen:

- Übermäßiger Sammlungsaufwand — Stetig steigende `unusedStorageSize` Kennzahlen bei umfangreichen Schreibvorgängen oder Massenlöschungen, insbesondere bei großen Indizes.
- Hoher Prozentsatz an toten Fragmenten — `documentFragmentStats` zeigt konstant hohe `deadDocFragmentsPercent` Werte (über 10-15%).
- Niedrigere Abfragelatenz — Erhöhte Abfragelatenz aufgrund der Häufung toter Dokumente.
- Verlängerte GC-Dauer — Garbage Collection-Operationen dauern länger als historische Durchschnittswerte. `gcRuntimeStats`
- Erhöhte GC-Verarbeitung — Hoch, `LongestActiveGCRuntime` was bedeutet, dass der Garbage Collector nicht mit den Systemanforderungen Schritt halten kann.

Beeinträchtigt die Garbage Collection die Leistung meiner Datenbank?

Unter normalen Bedingungen hat die Garbage Collection nur minimale Auswirkungen auf die Leistung. Wenn die Garbage Collection jedoch ins Hintertreffen gerät, kann Folgendes auftreten:

- Erhöhte Speicherkosten aufgrund angesammelter toter Dokumente.
- Langsamere Abfrageleistung aufgrund veralteter Indexeinträge.

- Temporärer schreibgeschützter Modus, wenn MVCC-IDs erschöpft sind.
- Höherer Ressourcenverbrauch bei intensiven Erfassungsläufen, insbesondere bei kleineren Instances.
- Geringere Effizienz bei Vorgängen im erweiterten Speichersegment für große Dokumente.

Kann ich die Müllabfuhr manuell auslösen?

Nein, die Müllabfuhr in Amazon DocumentDB kann nicht manuell ausgelöst werden. Das System verwaltet die Müllabfuhr im Rahmen seiner internen Wartungsarbeiten automatisch.

Welche Alarme sollte ich als Best Practice für den Betrieb einrichten?

Richten Sie die Überwachung sowohl auf Cluster- als auch auf Sammlungsebene ein, um eine optimale Leistung Ihres Amazon DocumentDB-Systems sicherzustellen.

Für die Überwachung auf Cluster-Ebene erstellen Sie zunächst einen CloudWatch Amazon-Alarm für die `AvailableMVCCIds` Metrik mit einem Schwellenwert von 900 Millionen bei Amazon DocumentDB 5.0 und höher oder 1,3 Milliarden bei Amazon DocumentDB 3.6 und 4.0. Dies gibt Ihnen ausreichend Zeit, um Maßnahmen zu ergreifen, bevor die Metrik Null erreicht. An diesem Punkt würde Ihr Cluster in den schreibgeschützten Modus wechseln. Beachten Sie, dass diese Metrik je nach Ihren spezifischen Nutzungsmustern schwanken kann. Einige Kunden sehen, dass sie unter den Schwellenwert fällt und sich dann wieder auf über 1,5 Milliarden erholt, wenn die Garbage Collection ihre Arbeit abgeschlossen hat.

Es ist auch wichtig, die `LongestActiveGCRuntime` Metrik über Amazon CloudWatch zu überwachen. Diese Metrik hilft Ihnen außerdem zu verstehen`gcRuntimeStats`, wie effizient die Speicherbereinigung in Ihrem System funktioniert.

Konzentrieren Sie sich bei der Überwachung auf Sammlungsebene auf die folgenden wichtigen Kennzahlen:

- `MVCCIdScale`— Achten Sie auf steigende Werte, die darauf hindeuten, dass MVCC-IDs älter werden und möglicherweise Aufmerksamkeit erfordern.
- `gcRuntimeStats`— Identifizieren Sie Garbage Collection-Prozesse, die ungewöhnlich lange dauern oder sich über mehrere Tage erstrecken.
- `documentFragmentStats`— `deadDocFragmentsPercent` Werte überwachen — konstant hohe Prozentsätze (über 10-15%) können darauf hindeuten, dass die Müllabfuhr ins Hintertreffen gerät.

- `storageSizeStats`und `unusedStorageSize` — Verfolgen Sie die Speicherauslastungsmuster und identifizieren Sie Sammlungen mit erheblichem ungenutztem Speicherplatz in beiden Speichersegmenten.

Sammlungen mit häufigen Schreibvorgängen erfordern besondere Aufmerksamkeit, da sie dem Garbage Collector mehr Arbeit abverlangen. Überprüfe diese Metriken häufiger für Sammlungen mit hoher Schreibaktivität, um sicherzustellen, dass die Garbage Collection mit deiner Arbeitslast Schritt hält.

Beachten Sie, dass diese Überwachungsempfehlungen als Ausgangspunkt dienen. Wenn Sie sich mit dem Verhalten Ihres Systems vertraut machen, sollten Sie diese Schwellenwerte möglicherweise anpassen, um Ihren spezifischen Nutzungsmustern und Anforderungen besser gerecht zu werden.

Was muss ich tun, wenn meine **`verfügbaren MVCCIDs den Alarmschwellenwert`** unterschreiten?

Wenn Ihre `AvailableMVCCIds` Metrik unter den Alarmschwellenwert fällt, ergreifen Sie sofort Maßnahmen, um zu verhindern, dass Ihr Cluster in den schreibgeschützten Modus wechselt. Skalieren Sie zunächst Ihre Instance-Größe, um dem Garbage Collector mehr Rechenressourcen zur Verfügung zu stellen. Auf diese Weise kann Ihre Anwendung ihren normalen Betrieb fortsetzen und gleichzeitig erhält der Garbage Collector die zusätzliche Leistung, die er benötigt, um den Anschluss aufzuholen.

Wenn eine Skalierung allein die Situation nicht verbessert, sollten Sie erwägen, Ihre Schreibvorgänge zu reduzieren. Verwenden Sie die `MVCCIdScale` Metrik, um zu ermitteln, welche spezifischen Sammlungen ältere MVCC-IDs enthalten, die Aufmerksamkeit erfordern. Überwachen Sie außerdem, `documentFragmentStats` um Sammlungen mit einem hohen Prozentsatz an toten Fragmenten zu identifizieren, die zur Ineffizienz der Speicherbereinigung beitragen könnten.

Sobald Sie diese Sammlungen identifiziert haben, müssen Sie möglicherweise vorübergehend die Schreibvorgänge auf sie reduzieren, damit die Garbage-Collection aufholen kann. Überwachen Sie die `AvailableMVCCIds` Kennzahl während der Erholungsphase genau, um sicherzustellen, dass Ihre Aktionen die gewünschte Wirkung haben. Ihr Cluster gilt als intakt, sobald der `AvailableMVCCIds` Wert auf 1,5 Milliarden oder höher zurückkehrt.

Denken Sie daran, dass es sich bei diesen Schritten um vorbeugende Maßnahmen handelt, mit denen Ihr System wiederhergestellt werden kann, bevor es einen kritischen Zustand erreicht. Je früher Sie Maßnahmen ergreifen, nachdem die Metrik unter den Schwellenwert gefallen ist, desto wahrscheinlicher ist es, dass Sie jegliche Auswirkungen auf Ihre Schreibvorgänge vermeiden.

Wirken sich Batch-Operationen auf die MVCC-ID-Verbrauchsrate aus?

Ja. Da ein Batch, der innerhalb einer einzelnen atomaren Transaktion ausgeführt wird, nur eine MVCC-ID verbraucht, verringert die Gruppierung von Schreibvorgängen in Batches die Rate, mit der MVCC-IDs verbraucht werden. Es hat sich bewährt, den MVCC-ID-Verbrauch so gering wie möglich zu halten `insertMany` und `updateMany` die Belastung durch die Garbage Collection zu reduzieren.

Warum werden **`verfügbare MVCCIDs erst`** wiederhergestellt, wenn ein bestimmter Schwellenwert erreicht ist?

Für einen Workload `AvailableMVCCIds` mit hohem Schreibzugriff, der noch nicht genug MVCC-IDs verbraucht hat, um den Speicherbereinigungszyklus auszulösen, wird vorübergehend ein stetiger Rückgang ohne Wiederherstellung erwartet. Die Speicherbereinigung wird ausgelöst, wenn eine der folgenden Bedingungen erfüllt ist: ein bestimmter Teil der Dokumente innerhalb einer Sammlung ist veraltet, ein bestimmter Teil neuer Dokumente wird eingefügt oder der Schwellenwert für die Rückgewinnung wird `AvailableMVCCIds` unterschritten.

Solange der Schwellenwert für die Rückgewinnung nicht `AvailableMVCCIds` unterschritten wird, findet keine erzwungene Reklamation statt. Sobald dies der Fall ist, wird eine Reklamation ausgelöst und wiederhergestellt. `AvailableMVCCIds` Langfristige Transaktionen und offene Cursor können jedoch den Rückforderungshorizont genau bestimmen und eine Reklamation verhindern, selbst wenn der Schwellenwert erreicht ist.

Fehlercodes

Wenn ein Befehl fehlschlägt, erhalten Sie eine Fehlermeldung von Amazon DocumentDB (mit MongoDB-Kompatibilität), die einen numerischen Fehler `code` und eine beschreibende Meldung () enthält. `errmsg` Identifizieren Sie anhand der folgenden Tabelle die Fehlercodes, die Amazon DocumentDB 8.0 möglicherweise an Ihre Client-Anwendung zurückgibt. Wenn ein Fehlercode einer einzelnen Bedingung entspricht, entspricht die Beschreibung dem vom Server zurückgegebenen Meldungstext. Codes, die mehrere Bedingungen abdecken, haben eine allgemeine Beschreibung.

> **Important**
>
> Erstellen Sie keine Anwendungslogik, die von bestimmten numerischen Fehlercodes abhängt, da Amazon DocumentDB Fehlercodes zwischen Hauptversionen ändern könnte.

Fehlercode	Beschreibung
1	Artikel wurde nicht gefunden
2	Ein Befehlsargument, eine Option oder ein Feldwert ist ungültig oder wird nicht unterstützt.
4	Ein erforderliches Feld fehlt oder ein Konfigurationswert ist nicht richtig formatiert.
5	Eine View-Definition verweist auf sich selbst und erzeugt einen Zyklus, oder es ist ein Dateisystemfehler aufgetreten.
9	Ein Befehls-, Feld-, Hinweis- oder Resume-Token konnte nicht analysiert werden oder hat ein ungültiges Format.
10	Die Zielsammlung wurde nicht gefunden oder es wurde keine Sammlung angegeben.
11	Ein Benutzer, auf den verwiesen wird, konnte nicht gefunden werden, oder die Zielsammlung ist bereits vorhanden.
13	Sie sind nicht berechtigt, diesen Befehl auszuführen, oder Ihre Authentifizierung ist ungültig.
14	Ein Feld oder Wert hat den falschen Datentyp.
15	Ein Wert hat während der Auswertung seinen zulässigen Bereich überschritten.
16	Der Schreibbatch ist leer, falsch formatiert oder überschreitet die zulässige Anzahl an Vorgängen.
17	Nicht alle Objekte haben den gleichen Typ
18	Die Authentifizierung ist fehlgeschlagen, weil die Anmeldeinformationen ungültig sind oder der Benutzer nicht existiert.
19	Der primitive Datenverarbeitungsvorgang wurde nicht gefunden oder der primitive Typ wurde nicht gefunden
20	Die angeforderte Operation ist in diesem Zusammenhang nicht zulässig.

Fehlercode	Beschreibung
26	Der Ziel-Namespace oder die Zielsammlung ist nicht vorhanden.
27	Der angeforderte Index existiert nicht oder es ist kein geeigneter Index verfügbar.
37	Ein Argument für die Pipeline oder Aggregationsphase fehlt, ist leer oder ungültig.
38	Der Datenbankname wurde nicht gefunden
39	Der Vorgang wurde abgebrochen, weil er MaxTimeMS überschritten hat, zu lange ausgeführt wurde oder der verfügbare Arbeitsspeicher überschritten wurde.
40	Ein Update steht in Konflikt mit einem anderen Feld, oder Schreibvorgänge werden durch die Garbage Collection gedrosselt oder deaktiviert.
41	JavaScript Fehler
43	Der angegebene Cursor existiert nicht mehr oder ist abgelaufen.
44	Der Array-Index liegt außerhalb der Grenzen
48	Eine Sammlung, Ansicht oder ein Namespace mit diesem Namen ist bereits vorhanden.
51	Ungültiger Aktualisierungsoperator
52	Ein Feldname verwendet ein reserviertes $-Präfix, das für die Speicherung nicht gültig ist.
53	Das Feld _id hat einen ungültigen oder nicht unterstützten Typ, z. B. ein Array oder eine Regex.
54	kann keine abzulegenden Abfragefelder ableiten
55	Die Transaktion wurde bereits bestätigt
56	Ein Feldname in der Anfrage ist leer, was nicht zulässig ist.
57	Die Transaktion wurde durch eine andere Transaktion für ungültig erklärt

Fehlercode	Beschreibung
58	Feldwert ist zu lang
59	Der angeforderte Befehl ist unbekannt oder wird nicht unterstützt.
61	Ungültige Sammlung
62	An der Sammlung können keine Änderungen vorgenommen werden
65	Batch-Operationsfehler sind aufgetreten
66	Ein Update hat versucht, das unveränderliche _id-Feld zu ändern.
67	Der Index konnte aufgrund einer ungültigen oder nicht unterstützten Spezifikation nicht erstellt werden.
68	Ein Index mit widersprüchlichen Eigenschaften ist bereits vorhanden.
70	Das mit $snippet verwendete Feld ist nicht als Snippet-Spalte konfiguriert
71	Der Index ist bereits mit verschiedenen Optionen vorhanden
72	Eine oder mehrere Optionen, die für den Befehl oder den Index angegeben wurden, sind ungültig.
73	Der Namespace-, Datenbank- oder Sammlungsname ist für diesen Vorgang ungültig oder nicht zulässig.
74	Für die Sammlung ist kein Textindex definiert.
75	Ein Textindex ist für diese Sammlung definiert, aber nicht für dieses Feld.
77	Snapshot kann nicht mit Sortierung verwendet werden
78	Dokument nicht gefunden
80	Die Abfrage überschreitet die maximal zulässige Größe.
82	Der Zeitrahmen hat eine ungültige Formatierung. <number>Gültiges Format: '1 m' oder '3 y' '//' h/d w/s m/y

Fehlercode	Beschreibung
85	Ein Index mit diesem Namen existiert bereits, aber mit anderen Optionen.
86	Der angeforderte Index steht in Konflikt mit einem vorhandenen Index, der denselben Namen hat.
87	Die $sort-Schlüsselreihenfolge muss 1 (für aufsteigend) oder -1 (für absteigend) sein
92	Nicht genug Platz für den Betrieb
93	Der Benutzer ist bereits vorhanden
94	Die Zielsammlung, der Cluster oder die Ressourcenberechtigung ist ungültig oder nicht verfügbar.
95	Die Datenbank ist nicht vorhanden
96	Der Parameter 'WriteConcern' ist ungültig
97	Das Passwort darf nicht leer sein
99	Undefinierte Rolle oder undefiniertes Privileg
102	Es kann keine Rolle hinzugefügt werden, die bereits integriert ist
103	Eine gültige Rolle konnte nicht aus der Sammlung extrahiert werden.
105	Der Benutzer ist ungültig oder die Autorisierungsbibliothek konnte nicht initialisiert werden.
107	Der Datenbank können keine Rollen aus einer anderen Datenbank zugewiesen werden
111	_id muss bei der Bereitstellung des Aktualisierungsdokuments genau angegeben werden
112	Konflikt schreiben
113	Die angegebene Rolle ist nicht vorhanden.

Fehlercode	Beschreibung
114	Die Rolle ist bereits vorhanden
115	AnyResource sollte den Wert true haben, wenn angegeben
116	Ressourcen sollten entweder db und collection set, cluster set oder anyResource set haben
117	Der Vorgang steht in Konflikt mit einem anderen laufenden Vorgang, z. B. einer aktiven Transaktion.
120	„ShowPrivileges“ kann nicht angegeben werden, da wir alle Benutzer anzeigen
121	Die Überprüfung des Dokuments ist fehlgeschlagen
122	Fehler beim Aktualisieren
124	Fehler beim Verarbeiten der Schema-ID
125	Der Befehl ist fehlgeschlagen, häufig aufgrund einer mehrdeutigen Referenz oder einer fehlenden Sitzung.
126	Die Datenbank ist bereits vorhanden
127	Der Datenbankname ist bereits in einer anderen Groß- und Kleinschreibung enthalten
128	Zu viele Felder
129	Ich warte auf das Löschen der Datenbank
130	Ungültiger Parameterwert
136	Die Cursorposition wurde verloren oder eine erforderliche Change-Stream-Funktion ist nicht aktiviert.
139	JS-Interpreter-Fehler
145	Alias ist in der Abfrage nicht definiert

Fehlercode	Beschreibung
147	nicht Meister
148	Nicht genügend Ressourcen, um die Abfrage auszuführen
149	Alias wurde bereits in der Abfrage definiert
154	Ein Argument hat einen unzulässigen Wert oder einen falschen (nicht numerisch en) Typ.
155	Ein NULL-Wert wurde gefunden
156	Es wurde ein UNDEFINIERTER Wert festgestellt
157	NAN-Wert angetroffen
158	Sammlung, die bereits für einen anderen Zweck verwendet wird
159	Ein Wert für die Dauer ist überschritten oder ein Netzwerkvorgang ist fehlgesch lagen.
161	Die angeforderte Kollationsversion ist ungültig oder nicht verfügbar.
163	Ein Feld oder Name ist für die Speicherung nicht gültig.
164	Ein Abfrageoperator wurde auf eine Weise verschachtelt, die nicht zulässig ist.
165	Eine Ansicht überschreitet die maximale Verschachtelungstiefe von 20 oder verweist in einem Zyklus auf sich selbst.
166	Die Operation wird für eine Ansicht nicht unterstützt, oder es wurden zu viele Positionselemente verwendet.
167	Die angeforderte Operation oder Phase wird in einer Ansicht nicht unterstützt.
168	Ein Ausdruck oder ein Pipelineoperator wurde nicht erkannt oder hat ein ungültiges Argument.
169	BadValue $maxTimeMS hat einen nicht ganzzahligen Wert

Fehlercode	Beschreibung
170	BadValue $maxTimeMS liegt außerhalb des zulässigen Bereichs
171	Ein Index kann nicht über mehrere Array-Felder erstellt werden.
173	Der Wert für Skip muss positiv sein
174	Die erwartete Ausführlichkeit des Protokolls ist eine Zahl
181	Das Schlüsselmuster entspricht mehreren Indizes; geben Sie den Index stattdessen nach Namen an.
197	Die Indexspezifikation enthält ein ungültiges oder nicht unterstütztes Feld.
207	uuid muss ein 16-Byte-Binärfeld mit dem Untertyp UUID (4) sein
224	Diese Abfragefunktion ist im aktuellen Kontext nicht zulässig.
225	In der Sitzung wurde eine neuere Transaktion gestartet, sodass diese ältere Transaktion nicht fortgesetzt werden kann.
241	Ein Wert konnte in $convert nicht in den angeforderten Typ konvertiert werden.
250	Ungültige Ansicht
251	Die angegebene Transaktionsnummer stimmt mit keiner aktiven Transaktion überein.
252	Rekursionslimit erreicht
253	Eintrag konnte nicht in die Ansicht eingefügt werden
254	Der Eintrag konnte nicht aus der Ansicht gelöscht werden
256	Das Dokument wurde nach der $happing-Klausel gefiltert
257	Die Gesamtgröße aller Operationen in der Transaktion überschreitet das zulässige Limit.

Fehlercode	Beschreibung
263	Dieser Vorgang kann nicht innerhalb einer Transaktion mit mehreren Dokumente n ausgeführt werden.
291	Der Index für die $geoNear-Abfrage konnte nicht gefunden werden
300	Die Authentifizierung ist aufgrund eines SASL-Handshakes oder eines Sitzungsf ehlers fehlgeschlagen.
301	Ein Authentifizierungsmechanismus oder ein angeforderter Vorgang wird nicht unterstützt.
302	Für die Authentifizierung sind zusätzliche Daten erforderlich, um den SASL-Aust ausch abzuschließen.
303	Die angeforderte Stufen-, Optionen- oder Rollenkonfiguration wird nicht unterstüt zt.
304	Die angeforderte Aggregationsphase wird nicht unterstützt.
305	Der Operator für die Aggregationsprojektion wird nicht unterstützt.
306	Der angegebene Aggregationsgruppen-Akkumulator-Operator wird nicht unterstüt zt.
502	Die Spalte ist bereits vorhanden
505	Einfügen ist hier nicht erlaubt
601	Die Koordinaten liegen außerhalb der Grenzen
602	Indexfehler
700	Änderungen an diesem Parameter sind im Live-Modus nicht zulässig
701	Ungültiger Wert für diesen Parameter
703	Unzulässiger Zustandsübergang für diesen Parameter
999	Schlechte Transaktions-ID

Fehlercode	Beschreibung
4570	$lookup-Parameter erwartet Zeichenkettenwert
4572	$lookup fehlen Pflichtfelder
10065	Einem Parameter, der ein Objekt sein muss (wie vars oder $rand), wurde ein Wert zugewiesen, der kein Objekt ist.
10107	nicht Meister
11000	Ein doppelter Schlüssel verstößt gegen eine eindeutige Indexeinschränkung.
11601	Die Operation wurde unterbrochen.
11603	Die Syntax der Client-Anfrage ist beschädigt oder ungültig
13103	zu viele zusammengesetzte Schlüssel
13297	Eine Datenbank mit demselben Namen existiert bereits, unterscheidet sich jedoch nur in der Groß- und Kleinschreibung.
13388	Unbekannter Pipeline-Stufenname: $collStats
13435	nicht master und slaveOK=False
14031	Nicht genug Speicherplatz
15888	muss den Namen der zu erstellenden Sammlung übergeben
15947	Die Felder einer Gruppe müssen in einem Objekt angegeben werden
15950	Der Name des Gruppenaggregatfeldes kann kein Operator sein
15952	unbekannter $group-Operator
15955	Syntaxfehler im Gruppenaggregationsoperator: Das Feld _id muss angegeben werden
15956	Das Argument $skip ist negativ; es muss eine nicht negative Zahl sein.
15957	das Limit muss als Zahl angegeben werden

Fehlercode	Beschreibung
15958	Der $limit-Wert muss eine positive Zahl sein.
15959	der Match-Filter muss ein Ausdruck in einem Objekt sein
15969	Syntaxfehler: Der Wert des $project-Operators muss ein Dokument sein.
15972	Das Argument für $skip muss eine Zahl sein
15973	Die $sort-Schlüsselspezifikation muss ein Objekt sein
15974	Die $sort-Schlüsselreihenfolge muss mit einer Zahl angegeben werden
15975	Die $sort-Schlüsselreihenfolge muss 1 (für aufsteigend) oder -1 (für absteigend) sein
15976	$sort stage muss mindestens einen Sortierschlüssel haben
15981	Das Abfragefeld ist keine Zeichenfolge
15983	Ein Objekt, das einen Ausdruck darstellt, muss genau ein Feld haben.
15992	unzulässiger Feldtyp NULL im Objektausdruck
15998	Feldpfadnamen dürfen keine leeren Zeichenfolgen sein.
16006	Ein Wert konnte nicht in ein Datum konvertiert werden, da es sich nicht um einen gültigen Datumstyp handelt.
16007	Ein Wert kann nicht in eine Zeichenfolge konvertiert werden, da eine Zeichenfo lge erwartet wurde.
16020	Ein Ausdrucksoperator wurde mit der falschen Anzahl von Argumenten aufgerufe n.
16034	Der Startindex von $substrBytes muss ein numerischer Typ sein.
16035	Das Längenargument $substrBytes muss ein numerischer Typ sein.
16395	$where ist innerhalb eines $match-Aggregationsausdrucks nicht zulässig

Fehlercode	Beschreibung
16400	kann keinen Ausdruck für ein Feld hinzufügen, da es bereits einen Ausdruck für dieses Feld oder eines seiner Unterfelder gibt.
16401	kann keinen Ausdruck für ein Unterfeld hinzufügen, da es bereits einen Ausdruck gibt, der für das gesamte Feld gilt
16404	$-Ausdrücke sind in diesem Kontext nicht erlaubt
16406	Andere Felder als _id können nicht ausgeschlossen werden
16410	Ein $project-Feldname oder -Pfad beginnt illegal mit '$' oder enthält '$'.
16411	Feldnamen dürfen keine Nullzeichen enthalten
16419	Feldpfad darf keine eingebetteten Nullzeichen enthalten
16424	$near ist innerhalb eines $match-Aggregationsausdrucks nicht zulässig
16426	$nearSphere/$NEAR ist innerhalb eines $match-Aggregationsausdrucks nicht zulässig
16435	Die Pipeline-Komponente muss ein Element enthalten
16493	Es wurde versucht, eine Zeichenfolge zu erstellen, die länger als 16 MB ist
16554	$add unterstützt nur numerische oder Datumstypen.
16605	$geoNear erfordert ein gültiges 'near'-Feld und korrekt eingegebene GeoNear-Optionen.
16606	$geoNear benötigt eine 'distanceField'-Option als String
16607	$geoNear erfordert, dass die angegebene Option als Zeichenfolge angegeben wird.
16608	Dividiere durch Null
16610	$mod erhielt einen Divisor von Null.

Fehlercode	Beschreibung
16611	$mod unterstützt nur numerische Typen für beide Operanden.
16612	In einem $add-Ausdruck ist nur ein Datum zulässig
16613	Bei der Operation wurde versucht, einen nicht numerischen Wert ohne Datum von einem Datum zu subtrahieren.
16702	$concat unterstützt nur Zeichenkettenargumente.
16730	Text IndexVersion ist außerhalb des zulässigen Bereichs
16755	Geometrie mit strenger Windungsreihenfolge kann nicht indexiert werden
16800	kann nicht 2 Geofelder haben
16801	2d muss an erster Stelle im Index stehen
16804	Standortobjekt erwartet, Standort-Array nicht im richtigen Format
16810	$mod benötigt ein Array mit genau zwei Werten als Parameter.
16837	Ein Aktualisierungspfad kann dasselbe Feld nicht mehrmals ändern.
16868	Ungültiger Variablenname
16869	Ungültiger Variablenname
16870	Ungültiger Variablenname
16872	Ein Feldpfad ist leer oder '$' allein wird als Feldpfad in $project verwendet.
16873	Fieldpath beginnt nicht mit $
16874	$let unterstützt nur ein Objekt als Argument
16875	$let hat einen unbekannten Parameter erhalten.
16876	Fehlender 'vars' -Parameter für $let
16877	Fehlender 'in'-Parameter für $let

Fehlercode	Beschreibung
16878	$map unterstützt nur ein Objekt als Argument
16879	$map erhält einen unbekannten Parameter.
16880	$map fehlt der erforderliche Eingabeparameter.
16881	Fehlendes erforderliches Feld
16882	Fehlender 'in'-Parameter für $map
16883	Die Eingabe für $map muss ein Array sein.
16898	$minDistance muss nicht negativ sein
16899	$maxDistance muss eine Zahl sein
16900	$maxDistance muss nicht negativ sein
16945	Speicherlimit für $group oder $lookup überschritten. Geben Sie die Option Zulassen einDiskUse:true , um sich anzumelden.
16949	Ungültige Aggregationsoption; allow DiskUse muss ein boolescher Wert sein.
16990	$out erfordert eine Zeichenfolge oder ein Objektargument, das das Ausgabeziel benennt.
16996	Das Einfügen für $out ist fehlgeschlagen
16997	$out ist fehlgeschlagen, weil sich die Indizes der Zielsammlung während der Verarbeitung geändert haben.
17040	ElementsTrueDas Argument von $all muss ein Array sein.
17041	ElementTrueDas Argument von $any muss ein Array sein.
17042	Beide Operanden von $set IsSubset müssen Arrays sein.
17043	Alle Operanden von $setUnion müssen Arrays sein.
17044	Ein Operand von $setEquals ist kein Array.

Fehlercode	Beschreibung
17045	$setEquals benötigt mindestens zwei Argumente.
17046	Beide Operanden von $set IsSubset müssen Arrays sein.
17047	Alle Operanden von $setIntersection müssen Arrays sein.
17048	Beide Operanden von $setDifference müssen Arrays sein.
17049	Beide Operanden von $setDifference müssen Arrays sein.
17053	Der Ausdruck von $redact sollte außer den Variablen $$KEEP, $$DESCEND und $$PRUNE nichts zurückgeben
17080	Fehlender 'if'-Parameter für Cond
17081	Fehlender 'thnen'-Parameter für Cond
17082	Fehlender 'else'-Parameter für Cond
17083	Unbekannter Parameter für $cond
17124	Das Argument für $size muss ein Array sein.
17275	ROOT kann nicht neu definiert werden
17276	Die Variable wurde nicht gefunden
17280	Der Indexschlüsselwert ist zu groß für die Indexierung.
17287	$merge kann dieselbe Zielzeile nicht mehr als einmal ändern.
17293	Die Textindexoption „TextIndexVersion“ muss eine Zahl sein
17302	Dieser Argumentwert muss größer oder gleich Null sein.
17304	$geoNear erfordert einen gültigen 'Near'-Punkt- und Geoabfrage-Argumente.
17308	$meta erhält ein Argument, das nicht unterstützt wird.
17310	Beide Operanden von $set IsSubset müssen Arrays sein.

Fehlercode	Beschreibung
17313	$match mit $text ist nur als erste Pipeline-Phase zulässig.
17320	CreateCollection kann nicht auf einem Namespace ausgeführt werden, der einen reservierten Namen enthält.
17385	Kann nicht in die Spezialsammlung übertragen werden
17390	$group unterstützt keine Ausdrücke im Inklusionsstil
17419	Das nach der Aktualisierung resultierende Dokument überschreitet die maximal zulässige Größe.
17420	Das zu aktualisierende Dokument ist größer als 16777216
18533	Das Formatfeld muss eine Zeichenfolge sein
18534	Unbekanntes Feld bis $date ToString
18535	Die Zeichenfolge im Datumsformat ist aufgrund eines nachfolgenden Prozentze ichens ungültig.
18536	Ungültiges $date-Uhrzeitformat ToString
18537	$date ToString kann eine Datumskomponente außerhalb des Bereichs 0-9999 nicht formatieren.
18627	Fehlender 'Format'-Parameter für $date ToString
18628	Fehlender 'Date'-Parameter für $date ToString
18629	falscher Argumenttyp
25278	$geoNear benötigt ein 'distanceField'-Argument
26823	Eine Operations-ID kann nicht innerhalb der verfügbaren Bits dargestellt werden.
28539	Das erste Argument für den Distinct-Befehl muss eine Zeichenfolge sein
28646	$Filter erwartet ein Objekt als Parameter

Fehlercode	Beschreibung
28647	Unerwartetes Feld für $filter
28648	$Filter benötigt ein Eingabefeld
28649	$Filter benötigt ein 'as'-Feld
28650	$Filter benötigt ein 'cond' Feld
28651	Array-Eingabe erforderlich
28656	$substrBytes: Ungültiger Bereich, Startindex ist ein Fortsetzungsbyte. UTF-8
28657	$substrBytes: Ungültiger Bereich, der Endindex befindet sich in der Mitte eines Zeichens. UTF-8
28664	$concatArrays unterstützt nur Array-Eingaben.
28667	Ein Ausdruck wie $slice, $range oder $index OfArray hat die falsche Anzahl von Argumenten erhalten.
28680	$abs wird auf einen Wert wie den Mindestwert lang angewendet, der nicht negiert werden kann.
28689	Ein Operatorargument muss ein Array sein, aber es wurde ein anderer Typ angegeben.
28690	Das zweite Argument für das Array ElemAt muss numerisch sein
28691	Das zweite Argument für das Array ElemAt muss durch eine 32-Bit-Ganzzahl dargestellt werden
28714	Das Argument von $sqrt muss größer oder gleich 0 sein
28724	Das erste Argument für $slice muss ein Array sein.
28725	Das zweite Argument für $slice muss ein numerischer Wert sein.
28726	Das zweite Argument für $slice muss in eine 32-Bit-Ganzzahl passen.

Fehlercode	Beschreibung
28727	Das dritte Argument für $slice muss eine Zahl sein.
28728	Das dritte Argument für $slice muss eine 32-Bit-Ganzzahl sein.
28729	Das dritte Argument von $slice muss eine positive 32-Bit-Ganzzahl sein.
28745	Die $sample -Stufenspezifikation muss ein Objekt sein
28746	Das Größenargument für $sample muss eine Zahl sein
28747	Das Größenargument für $sample darf nicht negativ sein
28748	$sample hat eine unbekannte Option erhalten.
28749	$sample stage muss eine Größe angeben
28756	Das Argument von $log muss numerisch sein.
28757	$log erfordert, dass sein Basisargument ein numerischer Wert ist.
28758	Das Argument für $log muss eine positive Zahl sein.
28759	Die Basis von $log muss eine positive Zahl ungleich 1 sein.
28761	Das Argument von $log10 muss eine positive Zahl sein.
28762	Die Basis von $pow muss ein numerischer Wert sein.
28763	Der Exponent von $pow muss numerisch sein.
28764	$pow kann keine Basis von 0 und einen negativen Exponenten annehmen
28765	$ceil, $exp, $floor, $ln oder $log10 erhält einen nichtnumerischen Typ.
28766	$ln erfordert ein positives Zahlenargument.
28812	Fieldpath ist leer
28818	Fieldpath beginnt nicht mit einem '$'

Fehlercode	Beschreibung
31022	Für diesen Operator muss ein Eingabeparameter angegeben werden.
31023	Für diesen Operator ist ein Regex-Parameter erforderlich.
31024	Der Operator erhält ein unbekanntes Argument.
31032	Ein Feldname darf kein eingebettetes Nullbyte enthalten.
31034	Ein Wert liegt außerhalb des zulässigen Bereichs von [-32768, 32767].
31095	WeekYear'iso' muss eine Ganzzahl zwischen 1 und 9999 ergeben.
31138	Der für die Sortierung verwendete $meta-Wert ist kein gültiges Metadaten-Schlüsselwort.
31170	Ein leeres Objekt wurde erwartet, aber ein nicht leerer Wert wurde bereitgestellt.
31249	Ein Projektionspfad kollidiert mit einem anderen Feld in der Projektion.
31254	Eine Inklusion kann nicht innerhalb einer Ausschlussprojektion angegeben werden.
31393	$bsonSize benötigt ein Dokument als Eingabe.
31394	Ein Operator für die Positionsprojektion kann nicht in der Mitte eines Feldpfads erscheinen.
34413	In einer Geonear-Abfrage wurde ein ungültiger Operator verwendet.
34435	Das Argument für $reverseArray muss ein Array sein.
34443	$range erfordert einen numerischen Startwert.
34444	$range erfordert einen Startwert, der als 32-Bit-Ganzzahl dargestellt werden kann.
34445	$range erfordert, dass der Endwert numerisch ist.
34446	Der Endwert von $range muss in eine 32-Bit-Ganzzahl passen.

Fehlercode	Beschreibung
34447	$range erfordert einen numerischen Schrittwert.
34448	$range erfordert einen Schrittwert, der als 32-Bit-Ganzzahl dargestellt werden kann.
34449	$range erfordert einen Schrittwert ungleich Null
34450	Der Startindex von $substrCp muss ein numerischer Typ sein.
34451	Der $substrCP-Startindex kann nicht als 32-Bit-Ganzzahl dargestellt werden.
34452	Das Längenargument für $substrCp ist kein numerischer Typ.
34453	Die Länge von $substrCp kann nicht als 32-Bit-Ganzzahl dargestellt werden.
34454	$substrCp: Die Länge muss eine nichtnegative Ganzzahl sein.
34455	$substrCp: Der Startindex muss eine nichtnegative Ganzzahl sein.
34460	$zip unterstützt nur ein Objekt als Argument.
34461	$zip-Eingaben müssen ein Array von Ausdrücken sein.
34462	Die Option 'Standardwerte' muss ein Array von Ausdrücken sein.
34463	Die LongestLength Verwendungsoption von $zip muss ein boolescher Wert sein.
34464	$zip erhielt ein unbekanntes Argument.
34465	$zip benötigt mindestens ein Eingabearray
34466	kann keine Standardwerte angeben, es sei denn, use LongestLength ist wahr
34467	Standardwerte und Eingaben müssen dieselbe Länge haben
34468	$zip empfängt in seiner Eingabe einen Ausdruck, der kein Array ist.
34471	$strlEncp erfordert ein String-Argument.
34473	$str LenBytes erfordert ein String-Argument.

Fehlercode	Beschreibung
40060	$switch benötigt ein Objekt als Argument.
40061	$switch erfordert, dass 'Branches' ein Array ist.
40062	$switch erfordert, dass jeder Zweig ein Objekt ist.
40063	$switch erhält ein unbekanntes Argument innerhalb eines Zweigs.
40064	$switch erfordert, dass jeder Zweig einen 'case-'-Ausdruck hat.
40065	$switch erfordert, dass jeder Zweig einen 'Then'-Ausdruck hat.
40066	$switch konnte keinen passenden Zweig für eine Eingabe finden und es wurde kein Standard angegeben.
40067	Der $switch-Ausdruck enthält ein unbekanntes Argument.
40068	$switch benötigt mindestens einen Zweig
40075	$reduce benötigt Objekt
40076	Unbekanntes Argument für $reduce
40077	$reduce benötigt 'Eingabe'
40078	$reduce benötigt 'initialValue'
40079	$reduce erfordert 'in'
40080	Die Eingabe für $reduce muss ein Array sein.
40081	Der Operator benötigt ein Array als zweites Argument.
40085	$split erfordert, dass sein erstes Argument zu einer Zeichenfolge ausgewertet wird.
40086	Das zweite Argument von $split muss zu einer Zeichenfolge ausgewertet werden.
40087	$split benötigt ein nicht leeres Trennzeichen

Fehlercode	Beschreibung
40090	$index OfArray benötigt ein Array als erstes Argument
40091	Das erste Argument für $index OfBytes ist kein String.
40092	$index OfBytes benötigt eine Zeichenfolge als zweites Argument.
40093	$indexOfCp benötigt eine Zeichenfolge als erstes Argument.
40094	$indexOfCp benötigt eine Zeichenfolge als zweites Argument.
40096	$index OfArray erfordert integrale Anfangs- und Endindexwerte.
40097	$index OfArray erfordert nichtnegative numerische Start- und Endindizes.
40148	das ByCount Sortierfeld muss als Pfad mit $-Präfix oder als Ausdruck innerhalb eines Objekts definiert werden
40149	das ByCount Sortierfeld muss als Zeichenfolge oder als Objekt angegeben werden
40156	das Zählfeld muss eine nicht leere Zeichenfolge sein
40157	das Zählfeld muss eine nicht leere Zeichenfolge sein
40158	Das Zählfeld darf kein Pfad mit dem Präfix $ sein
40159	das Zählfeld darf kein Null-Byte enthalten
40160	das Zählfeld darf '.' nicht enthalten
40169	Die $facet-Spezifikation muss ein nicht leeres Objekt sein
40170	Argumente für $facet müssen Arrays sein
40171	Elemente von Arrays in $facet spec müssen nicht leere Objekte sein
40172	Die Unterpipeline in der $facet-Stufe darf nicht leer sein
40176	Widersprüchliche Feldnamen angegeben

Fehlercode	Beschreibung
40177	$project und $addFields benötigen mindestens ein Ausgabefeld.
40178	Felder außer '_id' können in einer Inklusionsprojektion nicht ausgeschlossen werden
40179	Felder können nicht in eine Ausschlussprojektion aufgenommen, Felder zurückgesetzt oder berechnete Felder hinzugefügt werden
40180	$project erlaubt kein leeres Objekt als Feldwert.
40181	Ungültiges $project::, verursacht durch:: Eine Ausdrucksspezifikation muss genau ein Feld enthalten.
40183	Ein gepunkteter Feldname kann nicht in einem Unterobjekt verwendet werden.
40192	Das $bucket 'boundaries' Feld muss mindestens 2 Werte haben
40193	Alle Werte in der Option 'boundaries' für $bucket müssen denselben Typ haben
40194	Die Option 'boundaries' für $bucket muss sortiert werden, aber die Elemente sind nicht in aufsteigender Reihenfolge
40195	Das $bucket 'default' Feld muss ein konstanter Ausdruck sein
40196	Das $bucket 'output' Feld muss ein Objekt sein
40197	Unbekannte Option für $bucket
40198	$bucket erfordert die Angabe von 'groupBy' und 'boundaries'
40199	Das $bucket 'default' Feld muss kleiner als die niedrigste Grenze oder größer oder gleich der höchsten Grenze sein.
40200	Das $bucket 'boundaries' Feld muss ein Array sein
40201	Das Argument für die $bucket-Stufe muss ein Objekt sein
40202	Das $bucket-Feld 'groupBy' muss als Pfad oder Ausdruck mit dem Präfix $ definiert werden

Fehlercode	Beschreibung
40203	Das Feld muss ein Akkumulatorobjekt sein
40204	Der Feldname darf nicht '.' enthalten
40205	Der Feldname darf kein Operatorname sein
40206	Der Akkumulator ist ein unärer Operator
40218	Für die Abfrage sind Metadaten zur Textpunktzahl erforderlich, diese sind jedoch nicht verfügbar
40228	Der $replaceRoot-Ausdruck 'newRoot' muss zu einem Objekt ausgewertet werden.
40229	$replaceRoot benötigt ein Objekt als Spezifikation.
40230	$replaceRoot akzeptiert nur die Option 'newRoot'.
40234	Das Feld $group muss ein Akkumulatorobjekt sein.
40235	Ein Feldname darf kein '.'-Zeichen enthalten.
40236	Feldnamen dürfen nicht mit '$' beginnen
40237	Dieser Akkumulator ist ein unärer Operator und benötigt ein einzelnes Argument.
40239	Das $bucketAuto 'groupBy' Feld muss als Pfad mit $-Präfix oder als Ausdrucks objekt definiert werden
40240	Das Argument für $bucketAuto muss ein Objekt sein
40241	Das $bucketAuto 'bucket'-Feld muss ein numerischer Wert sein
40242	Das $bucketAuto 'bucket'-Feld muss als 32-Bit-Ganzzahl darstellbar sein
40243	Das $bucketAuto -Feld 'buckets' muss größer als 0 sein
40244	Das $bucketAuto 'output' Feld muss ein Objekt sein
40245	Unbekannte Option für $bucketAuto

Fehlercode	Beschreibung
40246	$bucketAuto erfordert die Angabe von 'groupBy' und 'buckets'
40257	Unbekannte Rundungsgranularität
40258	$bucketAuto kann eine 'Granularität' nur mit numerischen Grenzen angeben
40260	$bucketAuto kann eine 'Granularität' nur mit nicht negativen Zahlen angeben, hat aber eine negative Zahl gefunden
40261	Das $bucketAuto 'granularity' Feld muss eine Zeichenfolge sein
40272	Die $addFields-Stufenspezifikation muss ein Objekt sein.
40319	Die $lookup-Phasenspezifikation muss ein Objekt sein
40320	fehlende 'von'-Option in der $lookup-Phasenspezifikation
40321	Die Option 'from' für $lookup muss eine Zeichenfolge sein
40322	Ungültiger $look-Namespace
40323	Ein Spezifikationsobjekt für die Pipeline-Phase muss genau ein Feld enthalten.
40324	Die Pipeline enthält einen unbekannten Bühnennamen.
40331	Diese Aggregationsphase ist in einer $facet-Pipeline nicht zulässig.
40352	$project-Feldpfade dürfen keine leeren Zeichenfolgen sein.
40353	Ein Feldpfad darf nicht mit einem '.' enden.
40386	$array ToObject erfordert eine Array-Eingabe.
40390	$object ToArray benötigt ein Dokument als Eingabe.
40391	$array ToObject erfordert ein konsistentes Eingabeformat. Elemente müssen alle Arrays oder alle Objekte sein. Objekt wurde erkannt, jetzt gefunden: Array
40392	$array ToObject benötigt Objekte mit genau den Schlüsseln 'k' und 'v'.

Fehlercode	Beschreibung
40393	$array ToObject benötigt ein Objekt mit den Schlüsseln 'k' und 'v'. Es fehlen einer oder beide Schlüssel
40394	$array ToObject benötigt Objekte mit String-'k'-Schlüsseln und 'v'-Werten.
40395	$array ToObject benötigt Schlüssel-Wert-Paare, deren Schlüssel Zeichenketten sind.
40396	$array ToObject erfordert ein konsistentes Eingabeformat. Elemente müssen alle Arrays oder alle Objekte sein. Array wurde erkannt, jetzt gefunden: Objekt
40397	$array ToObject erfordert ein Array von Arrays mit zwei Elementen.
40398	$array ToObject erhält eine Eingabe eines unbekannten Typs oder Formats.
40400	$mergeObjects erfordert, dass alle Eingaben Objekte sind.
40414	Ein erforderliches Befehlsfeld fehlt oder die Überprüfung ist fehlgeschlagen.
40415	Der Befehl enthält ein unbekanntes Feld.
40422	Ein BSON-Array-Feldname hat einen ungültigen Wert.
40485	Eine unbekannte Zeitzonen-ID wurde angegeben.
40489	$date erlaubt es FromParts nicht, reguläre Kalenderfelder mit ISO-Woche ndatumsfeldern zu mischen
40515	Ein FromParts $date-Argument ergibt keine Ganzzahl.
40516	$date FromParts erfordert, dass entweder 'year' oder 'iso WeekYear 'vorhanden ist
40517	Die Option 'timezone' muss zu einer Zeichenfolge für Datumsoperatoren ausgewertet werden.
40518	$date FromParts oder $date ToParts haben einen unbekannten oder falsch eingegebenen Parameter erhalten.

Fehlercode	Beschreibung
40519	$date unterstützt FromParts nur ein Objekt als Argument.
40520	$date ToParts hat einen unbekannten Parameter erhalten.
40522	$date ToParts erfordert ein Objektargument mit einem 'Date'-Parameter.
40523	$date FromParts 'year' muss eine Ganzzahl im Bereich von 1 bis 9999 sein.
40525	Die FromParts week/day ISO-Werte von $date liegen außerhalb des gültigen Bereichs oder es fehlen Pflichtfelder.
40533	Das Zeitzonenargument muss eine Zeichenfolge sein.
40535	Der Stufe oder dem Ausdruck wurde eine unbekannte Option zugewiesen.
40536	Der Ausdruck akzeptiert genau ein Argument, wenn ein Array übergeben wird.
40539	Das erforderliche 'Datum'-Argument für einen Datumsoperator fehlt.
40540	$date unterstützt FromString nur ein Objekt als Argument
40541	$date FromString hat ein unbekanntes Argument erhalten.
40542	Fehlender 'DateString'-Parameter für $date FromString
40543	$date FromString erfordert, dass 'DateString' eine Zeichenfolge ist.
40545	Die date/time Zeichenfolge ist unvollständig und die erforderlichen Elemente fehlen.
40551	Sie können keine date/time Zeichenfolge mit Zeitzoneninformationen zusammen mit einem Zeitzonenargument übergeben
40553	$date FromString kann die angegebene Datumszeichenfolge nicht analysieren.
40554	Sie können keine date/time Zeichenfolge mit GMT-Offset zusammen mit einem Zeitzonenargument übergeben
40575	$changeStream 'FullDocument' muss „default“ oder „updateLookup“ sein.

Fehlercode	Beschreibung
40600	$out kann nicht innerhalb einer $facet-Stufe verwendet werden.
40601	$out und $merge können nur die letzte Phase in der Pipeline sein.
40602	Eine Phase wie $changeStream oder $geoNear ist nicht die erste Phase in der Pipeline.
40603	$geoNear war nach der Optimierung nicht die erste Phase in der Pipeline. Ist die Optimierung deaktiviert oder gehemmt?
40647	Das Resume-Token für den Change-Stream hat ein fehlendes oder falsch eingegebenes _data-Feld.
40674	Es ist nur ein Typ von Lebenslaufoption zulässig, es wurden jedoch mehrere gefunden.
50001	auto IndexId:false ist für die angegebene Sammlung nicht zulässig.
50694	Diesem Operator wurde ein unbekanntes Argument gegeben.
50695	Für diesen Operator ist ein Eingabefeld erforderlich.
50696	Der Operator unterstützt nur ein Objekt als Argument.
50699	Dieser Ausdrucksoperator erfordert, dass seine Eingabe eine Zeichenfolge ist.
50700	Das Argument 'chars' muss eine Zeichenfolge sein.
50736	GetMore kann in einer Sitzung nicht auf einem Cursor ausgeführt werden, der nicht in einer Sitzung erstellt wurde.
50737	Um GetMore auf einem in einer Sitzung erstellten Cursor auszuführen, muss eine LSID angegeben werden.
50752	Der $SubstrBytes-Startindex ist negativ.
50808	Die $changeStream-Stufe erwartet ein Dokument als Argument.
51024	Der Wert für die 'Stärke' der Sortierung muss 5 oder weniger betragen.

Fehlercode	Beschreibung
51047	Dieser Operator oder diese Stufe ist in einer $lookup-Subpipeline nicht zulässig.
51075	Optionen können nicht sowohl in $regex als auch in $options gesetzt werden
51076	Nicht genügend Speicher, um den Vektorindex für die angegebene Anzahl von Listen zu erstellen.
51078	Der Indexname konnte nicht gefunden werden.
51079	Latenzstatistiken werden nicht unterstützt
51091	Ein regulärer Ausdruck ist ungültig oder falsch geformt.
51103	Der Operator erwartet ein Objekt mit benannten Argumenten, hat aber einen anderen Typ erhalten.
51104	Der Operator erfordert, dass sein Eingabeargument eine Zeichenfolge ist.
51105	Das 'Regex'-Argument eines Operators ist kein String- oder Regex-Typ.
51106	Ein Operator erfordert, dass sein Parameter 'options' eine Zeichenfolge ist.
51107	$regex-Optionen können nicht sowohl in den Feldern 'Regex' als auch 'options' angegeben werden.
51108	Optionen für reguläre Ausdrücke enthalten ein ungültiges Flag oder ein eingebett etes Nullbyte.
51109	Ein regulärer Ausdruck kann kein eingebettetes Nullbyte enthalten.
51110	Optionen für reguläre Ausdrücke dürfen kein eingebettetes Nullbyte enthalten.
51111	Der für diesen Operator angegebene reguläre Ausdruck ist ungültig.
51151	Die Größe des Puffers zum Speichern der Ausgabe hat den Grenzwert für den Datentyp überschritten
51156	Ein regulärer Ausdruck konnte während der Auswertung in $regex nicht ausgeführt werden.

Fehlercode	Beschreibung
51270	Ein leeres Unterprojektionsobjekt ist kein gültiger Projektionswert.
51272	Syntaxfehler: $projection benötigt mindestens ein Ausgabefeld.
51744	Der Operator erfordert, dass 'replacement' eine Zeichenfolge ist.
51745	Der Operator erfordert, dass sein 'find'-Argument eine Zeichenfolge ist.
51746	Das Eingabeargument eines Operators ist keine Zeichenfolge.
51747	Ein Operator benötigt einen „Ersatz“ -Parameter.
51748	Für diesen Operator ist ein Find-Parameter erforderlich.
51749	Für diesen Operator ist ein Eingabeparameter erforderlich.
51750	Ein Aggregationsoperator hat ein unbekanntes Argument erhalten.
100003	Interner Systemfehler bei der Verarbeitung des optimierten Abfragepfads
100005	Ein konfiguriertes Limit, z. B. Sammlungen, benutzerdefinierte Rollen oder Rechte, wurde überschritten.
100006	Die maximale Anzahl zulässiger Benutzer wurde überschritten.
3040501	$rand akzeptiert derzeit keine Argumente
3041701	$getField hat ein unbekanntes Argument in $project erhalten.
3041702	Ungültiges $project::, verursacht durch:: $getField erfordert die Angabe von 'field'
3041703	Ungültiges $project::, verursacht durch:: $getField erfordert die Angabe von 'input'
3041704	$getField benötigt das Argument 'field', um eine Zeichenfolge auszuwerten.
4161100	$setField und $unsetField unterstützen nur ein Objekt als Argument.
4161101	$setField oder $unsetField wurde ein unbekanntes Argument übergeben.
4161102	$setField und $unsetField erfordern, dass ein 'Feld' angegeben wird.

Fehlercode	Beschreibung
4161103	Ungültiges $project::, verursacht durch:: $setField erfordert die Angabe von 'value'
416105	$setField oder $unsetField benötigen 'Eingabe', um ein Objekt auszuwerten.
4161106	$setField und $unsetField benötigen 'field', um als Konstante ausgewertet zu werden.
4161107	$setField oder $unsetField erfordert, dass 'field' zu einer Zeichenfolge ausgewert et wird.
4161108	$project verbietet hier eine Feldpfadreferenz; verwende stattdessen {$literal:...}.
4161109	$setField und $unsetField erfordern, dass ein 'Eingabefeld' angegeben wird.
4662500	Der Befehl enthält ein unbekanntes Feld.
4862100	Der vollqualifizierte Namespace überschreitet die maximal zulässige Länge.
5166301	$dateDiff unterstützt nur ein Objekt als Argument
5166302	$dateDiff erhielt ein unbekanntes Argument.
5166303	Fehlender 'StartDate'-Parameter für $DateDiff
5166304	Fehlender 'EndDate'-Parameter für $DateDiff
5166305	Fehlender 'Unit'-Parameter für $dateDiff
5166307	Ein $dateDiff-Argument ist kein Datum.
5166308	$DateDiff ist bei der Berechnung der Datumsdifferenz übergelaufen.
5166400	$dateAdd und $dateSubtract setzen voraus, dass ihr Argument ein Objekt ist.
5166401	$dateAdd oder $dateSubtract haben ein unbekanntes Argument erhalten.
5166402	$dateAdd oder $dateSubtract setzen voraus, dass StartDate, Einheit und Betrag vorhanden sind.

Fehlercode	Beschreibung
5166403	$dateAdd erfordert, dass StartDate in ein Datum umgewandelt werden kann
5166404	$dateAdd erwartet eine Zeichenfolge, die die Zeiteinheit definiert
5166405	$dateAdd erwartet eine ganzzahlige Anzahl von Zeiteinheiten
5166406	DateAdd ist übergelaufen
5166503	Ein unbekannter Wert für die Zeiteinheit wurde angegeben.
5439002	$dateTrunc ist bei der Berechnung des gekürzten Datums übergelaufen.
5439006	DateTrunc unterstützt keinen binSize-Wert
5439007	$dateTrunc unterstützt nur ein Objekt als Argument
5439008	Unbekanntes $dateTrunc-Argument; erwartetes Datum, Einheit, BinSize, Zeitzone, Start. OfWeek
5439009	Fehlender 'Date'-Parameter für $dateTrunc
5439010	Fehlender 'Unit'-Parameter für $dateTrunc
5439012	$dateTrunc erfordert, dass 'date' ein Datum ist.
5439013	$dateDiff und $dateTrunc erfordern, dass 'unit' eine Zeichenfolge ist.
5439014	Die 'Unit'-Werte von $dateDiff und $dateTrunc sind keine anerkannte Zeiteinheit.
5439015	$dateDiff oder $dateTrunc erfordern, dass 'start' eine Zeichenfolge ist. OfWeek
5439016	Der OfWeek 'start'-Wert von $dateDiff oder $dateTrunc ist kein gültiger Wochentag.
5439017	$dateTrunc erfordert, dass 'binSize' eine 64-Bit-Ganzzahl ist.
5439018	$dateTrunc erfordert, dass 'binSize' größer als 0 ist.
5447000	Die $collStats-Stufe muss ein verschachteltes Objekt sein.

Fehlercode	Beschreibung
5626500	$geoNear, $near und $nearSphere sind in diesem Zusammenhang nicht zulässig, da diese Operatoren das Sortieren von Geodaten erfordern. Wenn Sie keine Sortierung benötigen, sollten Sie stattdessen $geoWithin verwenden.
5654600	$project erlaubt in diesem Kontext keine Feldpfadreferenz.
5654601	Ungültiges $project:: verursacht durch:: $getField erfordert, dass 'field' als Konstante ausgewertet wird, hat aber ein Argument, das nicht konstant ist
5654602	$getField benötigt 'field', um eine Zeichenfolge auszuwerten.
5733415	kann $mod nicht um Null
5860400	$geoNear benötigt ein 'near'-Argument
5887502	Alle Operanden von $setEquals müssen Arrays sein.
5911200	PlanExecutor Fehler bei der Aggregation:: verursacht durch:: $mergeObjects unterstützt nur Objekte
5976500	$dateAdd wurde ein ungültiger 'Amount'-Parameterwert zugewiesen.
6045000	$dateSubtract hat einen ungültigen 'Amount'-Parameterwert erhalten.
7158303	$switch hat keinen passenden Zweig für die Eingabe gefunden und es wurde kein Standard angegeben.

Versionshinweise

Diese Versionshinweise beschreiben die Funktionen, Verbesserungen und Bugfixes von Amazon DocumentDB, geordnet nach Hauptversionen der Engine. In jedem Versionsabschnitt werden die Einträge nach Datum in einem Tabellenformat aufgelistet.

Einzelheiten zur Nummerierung von Engine-Versionen und Patch-Versionen (einschließlich der Zuordnung zwischen Engine-Patch-Versionspräfixen und Amazon DocumentDB-Engine-Versionen) finden Sie unter. Versionsnummerierung der Engine

Sie können die aktuelle Patch-Version der Amazon DocumentDB-Engine ermitteln, indem Sie den folgenden Befehl ausführen:

```
db.runCommand({getEngineVersion: 1})
```

> **Note**
>
> Ab Amazon DocumentDB 5.0.2 und 8.0.2 gibt dieser Befehl die Engine-Version und die Engine-Patch-Version als einzelne Zeichenfolge im Formular zurück. *`engine-version`*`+`*`engine-patch-version`* Beispielsweise kehrt ein Cluster auf Amazon DocumentDB 5.0.2 zurück`5.0.2+3.0.12345`, und ein Cluster auf 8.0.2 kehrt zurück. `8.0.2+4.0.12345` Cluster in früheren Versionen geben die Engine-Patch-Version eigenständig zurück, z. B. `3.0.12345` Weitere Informationen finden Sie unter Versionsnummerierung der Engine.

Wenn auf Ihrem Cluster nicht die neueste Version der Engine installiert ist, stehen wahrscheinlich Wartungsarbeiten zur Aktualisierung Ihrer Engine aus. Weitere Informationen finden Sie Wartung von Amazon DocumentDB im Entwicklerhandbuch.

Themen

- Versionshinweise zur Amazon DocumentDB-Engine
- Allgemeine Versionshinweise zu Amazon DocumentDB
- Versionshinweise zu Amazon DocumentDB Elastic Clusters

Versionshinweise zur Amazon DocumentDB-Engine

Versionshinweise für Versionen der Amazon DocumentDB-Engine, sortiert nach Hauptversionen.

Amazon DocumentDB 8.0

Versionshinweise für die Amazon DocumentDB-Engine-Version 8.0 (MongoDB 8.0-Kompatibilität).

Amazon DocumentDB 8.0.2

Date	Versionshinweise
28. September 2026	Amazon DocumentDB 8.0.2 (Maschinenversion: 4.0.12042) Neue Features und Verbesserungen 1. In Planner-Version 3 wurde Unterstützung für die `$setWindowFields` Aggregationsphase und ihre Fensteroperatoren hinzugefügt. 2. Unterstützung für die `$bucketAuto` Aggregationsphase in Planner-Version 3 hinzugefügt. 3. Unterstützung für die `$facet` Aggregationsphase in Planner Version 3 hinzugefügt. 4. Unterstützung für die `$graphLookup` Aggregationsphase in Planner Version 3 hinzugefügt. 5. Unterstützung für korrelierte `$lookup` Unterabfragen, einschließlich der präzisen korrelierten Unterabfragesyntax, mit einer Teilmenge von Pipeline-Stufen in Planner-Version 3 hinzugefügt. 6. Unterstützung für die Aggregationsphase hinzugefügt. `$changeStreamSplitLargeEvent` 7. Change-Streams geben jetzt Ereignisse für `createCollection` Operationen aus. 8. Change-Streams geben jetzt Ereignisse für `createIndex` Operationen aus. 9. Die Schreibleistung von Change-Streams wurde verbessert. 10. Unterstützung für wiederholbare Schreibvorgänge hinzugefügt.

Date	Versionshinweise
	11. Unterstützung für den `reIndex` Befehl für partielle Indizes hinzugefügt. 12. Planner Version 3 unterstützt jetzt Indexscans von `$expr` Prädikaten. 13. Planner Version 3 führt Scans vom Typ „Nur Index“ mit mehreren Feldern (IXONLYSCAN) für Suchabfragen ein. Weitere Informationen finden Sie unter Index-only Scans (abgedeckte Abfragen). 14. In Planner Version 3 werden nur Index-Scans (IXONLYSCAN) für Aggregationsphasen eingeführt, z. B. `$group` Weitere Informationen finden Sie unter Index-only Scans (abgedeckte Abfragen). 15. Leistungsoptimierung für `$count` und `countDocuments()` in Planner-Version 3. 16. Unterstützung für inkrementelles Sortieren in Planner-Version 3 hinzugefügt. Weitere Informationen finden Sie unter Inkrementelle Sortierung. 17. Verbesserte Planauswahl für Textindexabfragen. 18. Der Explain-Plan für Coalesced `$lookup` und `$unwind` Stages spiegelt jetzt den `includeArrayIndex` Parameter wider. 19. Unterstützung für Indexfilter für alle Befehle hinzugefügt. 20. Der Abfrageplaner priorisiert jetzt eindeutige Indizes für Equality (`$eq` -Prädikate. 21. Das Kürzungslimit für Aggregatbefehle in der Ausgabe wurde erhöht. `currentOp` Fehlerbehebungen 1. Ein Fehler im Abfrageplaner der Version 1 (Standard) wurde behoben. 2. Beim Major Version Upgrade (MVU) wurden Textumschläge für teilweise Indexausdrücke behoben. 3. Ein Profiler-Problem bei Abfragen mit schneller Zählung wurde behoben.

Date	Versionshinweise
	4. Sicherheitsupdate für den `collStats` Befehl.

Amazon DocumentDB 8.0.1

Date	Versionshinweise
15. September 2026	Sicherheits- und Stabilitätsverbesserungen Amazon DocumentDB 8.0.1 (Engine-Version: 4.0.11638) [in Bearbeitung] Amazon DocumentDB wendet diesen Patch automatisch während Ihres nächsten Wartungsfensters an.
31. August 2026	Neues Feature Amazon DocumentDB unterstützt jetzt direkte Hauptversions-Upgrades (MVU) direkt von den Engine-Versionen 3.6 und 4.0 auf Version 8.0. Dadurch sind keine Zwischenversions-Upgrades erforderlich und Sie können Ihre 3.6- oder 4.0-Cluster aktualisieren und dabei die vorhandenen Daten, Konfigurationen und Cluster-Einstellungen beibehalten. Hauptversions-Upgrades von Amazon DocumentDB 3.6 und 4.0 auf 8.0 sind in allen AWS Regionen verfügbar, in denen diese Versionen derzeit unterstützt werden. Weitere Informationen finden Sie unter Direktes Upgrade der Hauptversion von Amazon DocumentDB.
18. August 2026	Sicherheits- und Stabilitätsverbesserungen Amazon DocumentDB 8.0.1 (Engine-Version: 4.0.9938) Dieser Engine-Patch bietet die folgenden Verbesserungen: • Verbesserte Anforderungsverarbeitung Amazon DocumentDB validiert eingehende Client-Anfragen jetzt gründlicher entlang des Verbindungs- und Nachrichtenanalysepfads. Dies verbessert die

Date	Versionshinweise
	Widerstandsfähigkeit und Stabilität bei einer breiteren Palette von Eingabemustern. • Härtung des Authentifizierungs-Handshakes Amazon DocumentDB validiert den Authentifizierungsaustausch jetzt gründlicher. Dies verbessert die Robustheit beim Umgang mit unerwarteten oder falsch formatierten Eingaben. Amazon DocumentDB wendet diesen Patch automatisch während Ihres nächsten Wartungsfensters an.
11. August 2026	Neue Features Amazon DocumentDB 8.0.1 (Maschinenversion: 4.0.8864) • Unterstützung für Authentifizierung aktiviert. SCRAM-SHA-256 • Einstellbares RPO für globale Amazon DocumentDB-Cluster aktiviert. • Die Speicherrückgewinnung bei der Speicherbereinigung im Hintergrund wurde verbessert, um ungenutzten Speicherplatz im Laufe der Zeit zu reduzieren.
22. Juli 2026	Neues Feature Amazon DocumentDB-Cluster, auf denen Engine Version 8.0 und höher ausgeführt wird, unterstützen jetzt ein maximales Speicherlimit von 256 TiB, womit das bisherige Limit von 128 TiB verdoppelt wurde. Weitere Informationen finden Sie unter Amazon DocumentDB-Kontingente.

Date	Versionshinweise
13. Juli 2026	Amazon DocumentDB 8.0.1 (Maschinenversion: 4.0.8864) Neue Features und Verbesserungen 1. Unterstützung für 13 Akkumulator-Operatoren im Standard-Planner (v3):`$top`,,,`$bottom`,,`$topN`,`$bottomN`,,`$count`,`$firstN`,`$lastN`,`$minN` `$maxN` `$median` `$percentile` `$stdDevPop` `$stdDevSamp` 2. Unterstützung für 36 Ausdrucksoperatoren im Standard-Planner (v3): `$median` `$percentile` `$stdDevPop` `$stdDevSa mp` ,`$round`,`$sigmoid`,`$trunc`,`$firstN`,`$lastN`,`$maxN`,`$minN`,`$bitAnd ze` ,`$bsonSize` ,`$tsIncrement` ,`$tsSecond` ,`$cos`,`$acos`,`$cosh`,`$acosh`,`$sin`,`$asin`,`$sinh`,`$asinh`,`$tan`,`$atan`,`$ oRadians` , `$radiansToDegrees` `$isNumber` ,`$toUUID`. 3. Unterstützung für `$sampleRate` in `$match` im Standard-Planner (v3). 4. Unterstützung für `$sortByCount` den standardmäßigen Planner (v3). 5. Unterstützung für die `$listSearchIndexes` Aggregati onsphase im Standard-Planner (v3) hinzugefügt. Gibt Suchindex-Metadaten (Name, Status, abfragbares Objekt, Version) für Text- und Vektorsuchindizes für eine Sammlung zurück. 6. Unterstützt schnelle Komprimierung mithilfe des Nachrichtentyps OP_COMPRESSED. 7. Unterstützt die Methoden `min` und `max` cursor für den Befehl find im Standard-Planner (v3). 8. Planner Version 3 führt Index-Only-Scans (IXONLYSCAN) für ein. `countDocuments()` 9. Unterstützung für versteckte Indizes für alle Indextypen hinzugefügt (Planner Version 2 und höher). Fehlerbehebungen

Date	Versionshinweise
	1. Es wurde ein Problem behoben, bei dem `$text` Suchanfragen, die Unterstriche enthielten, falsche Ergebnisse lieferten, wenn sie als Restfilter ausgewertet wurden.

Amazon DocumentDB 8.0.0

Date	Versionshinweise
15. September 2026	Sicherheits- und Stabilitätsverbesserungen Amazon DocumentDB 8.0.0 (Engine-Version: 4.0.11440) [in Bearbeitung] Fehlerbehebungen Diese Version enthält die folgende Fehlerkorrektur: 1. Es wurde ein Problem behoben, bei dem `$text` Suchanfragen, die Unterstriche enthielten, falsche Ergebnisse lieferten, wenn sie als Restfilter ausgewertet wurden. Amazon DocumentDB wendet diesen Patch automatisch während Ihres nächsten Wartungsfensters an.
18. August 2026	Sicherheits- und Stabilitätsverbesserungen Amazon DocumentDB 8.0.0 (Engine-Version: 4.0.9912) Dieser Engine-Patch bietet die folgenden Verbesserungen: • Verbesserte Anforderungsverarbeitung Amazon DocumentDB validiert eingehende Client-Anfragen jetzt gründlicher entlang des Verbindungs- und Nachrichtenanalysepfads. Dies verbessert die Widerstandsfähigkeit und Stabilität bei einer breiteren Palette von Eingabemustern.

Date	Versionshinweise
	• Härtung des Authentifizierungs-Handshakes Amazon DocumentDB validiert den Authentifizierungsaustausch jetzt gründlicher. Dies verbessert die Robustheit beim Umgang mit unerwarteten oder falsch formatierten Eingaben. Wenn Sie ihn nicht früher anwenden, wendet Amazon DocumentDB diesen Patch automatisch während Ihres nächsten Wartungsfensters 30 Tage nach der Veröffentlichung an.
22. Juli 2026	Neues Feature Amazon DocumentDB-Cluster, auf denen Engine Version 8.0 und höher ausgeführt wird, unterstützen jetzt ein maximales Speicherlimit von 256 TiB, womit das bisherige Limit von 128 TiB verdoppelt wurde. Weitere Informationen finden Sie unter Amazon DocumentDB-Kontingente.
13. Mai 2026	Neues Feature Amazon DocumentDB serverless ist jetzt für Amazon DocumentDB 8.0 verfügbar. Weitere Informationen finden Sie unter Serverlose Verwendung von Amazon DocumentDB.
6. Mai 2026	Bugfixes und andere Änderungen Amazon DocumentDB 8.0 (Engine-Version: 4.0.6940) Dieses Update bietet einen optionalen Engine-Patch mit den folgenden Verbesserungen: 1. Es wurde ein Fehler in der Speicherbereinigung behoben, der im Extremfall zum Absturz von Lesegeräten führen konnte.

Date	Versionshinweise
20. April 2026	Neues Feature In-place Ein Hauptversions-Upgrade (MVU) von Amazon DocumentDB 5.0 auf Amazon DocumentDB 8.0 ist jetzt verfügbar. Weitere Informationen finden Sie unter Direktes Upgrade der Hauptversion von Amazon DocumentDB.

Date	Versionshinweise
20. April 2026	Neue Features und Verbesserungen Amazon DocumentDB 8.0 (Engine-Versionen: 4.0.5995, 4.0.6473) Dies ist ein erforderlicher Engine-Patch, der die folgenden neuen Funktionen, Verbesserungen und Korrekturen enthält: 1. `$sortArray` Betreiber in`$project`: • Unterstützung für `$sortArray` Operator in `$project` Phase in der Aggregationspipeline für Planner Version 3. Der `$sortArray` Operator sortiert ein Array anhand seiner Elemente (für Skalarwerte) oder eines angegebenen Felds (für Dokumentarrays) und gibt das sortierte Array in aufsteigender (1) oder absteigender () Reihenfolge zurück. `-1` 2. Unterstützung für`$out`,`$collStats` , mit Planner v3`$indexStats` . `$geoNear` Mit Planner V3 unterstützt die `$out` Stage auch das Schreiben in Sammlungen in verschiedenen Datenbanken. 3. Die Aggregationspipeline wird jetzt zusammengeführt `$lookup` und in `$unwind` Stufen unterteilt, auch wenn die `$unwind` Phase den Parameter enthält. `includeArrayIndex` Bisher wurde diese Optimierung übersprungen, wenn sie vorhanden war. `includeArrayIndex` 4. Leistungsverbesserungen zur Verbesserung des Schreibdurchsatzes für db.r8g.8xlarge und größere Instances, die im Rahmen der Konfiguration ausgeführt werden. I/O-Optimized 5. Sie können jetzt die Metrik für Instances überwachen. `Connections (active)` Weitere Informationen finden Sie unter Instanzkontingente. 6. Unterstützung für `$dateToParts` und `$dateFromParts` Aggregationsoperatoren. 7. Command-level Leistungsmetriken sind jetzt CloudWatch für CRUD-Operationen (`find`,,,`insert`,`findAndModify` ,`update`,`delete`,`aggregate` ,`count`,`distinct`,`commitTransaction`)

Date	Versionshinweise
	`getMoreabortTransaction` , `AvgDuration` Exposure und `MaxConcurrent` Dimensionen für eine verbesserte Betriebsü berwachung verfügbar. `P100Duration` 8. Es wurde ein Fehler im `$replaceRoot` Verhalten behoben, bei dem `newRoot` Felder, die boolesche Werte von`true`/`false`oder numerische Werte von`0`/enthielten, im `1` Ausgabedokument fehlten. 9. Es wurde ein Fehler behoben`$indexOfArray` , bei dem gültige Abfragen Fehler zurückgaben, die darauf hinwiesen, dass sie ungültig waren. 10. Der Explain-Plan beinhaltet jetzt `docsExamined` Metriken in der `executionStats` Ausgabe. 11. Die Leistung der SASL-Authentifizierung wurde verbessert. 12. Optimierte Dokumentaktualisierungen, um den Speicheraufwand zu reduzieren, wenn die Indexwerte unverändert bleiben und Dokumente in vorhandene Seiten passen. 13. Unterstützung für `bypassDocumentValidation` in`$merge`. Alle Amazon DocumentDB 8.0-Cluster werden im nächsten Wartungsf enster automatisch auf diese Patch-Version aktualisiert.
26. März 2026	Neue Features Amazon DocumentDB unterstützt jetzt die neueste Generation speicheroptimierter Instances, db.r8g, die von Arm-based AWS Graviton4-Prozessoren angetrieben werden und eine bis zu 30% bessere Leistung als R6G-Instances bieten. Weitere Informationen finden Sie in diesem Handbuch Instanzklassen verwalten

Date	Versionshinweise
11. November 2025	Neue Features Amazon DocumentDB bietet jetzt vollständige Wire-Protokollkomp atibilität mit MongoDB 8.0. Amazon DocumentDB 8.0 verbessert die Abfrageleistung, sodass Sie Hochleistungsanwendungen zu geringere n Kosten erstellen können. Amazon DocumentDB 8.0 (Engine-Patch, Version 4.0.4448) • Unterstützung für MongoDB 8.0-API-Treiber hinzugefügt. • Es wurde Unterstützung für Planner Version 3 hinzugefügt, die Leistungsverbesserungen auf Operatoren in der Aggregationsphase ausdehnt. Außerdem werden Optimierungen der Aggregationspipeli ne und verschiedene Befehle unterstützt. • Unterstützung für wörterbuchbasierte Komprimierung durch den Zstandard-Komprimierungsalgorithmus hinzugefügt, wodurch das Komprimierungsverhältnis um das Fünffache verbessert wurde. • Neue Funktionen hinzugefügt: Amazon DocumentDB 8.0 unterstützt Sortierung und Ansichten. • Unterstützung für neue Aggregationsstufen und Operatoren hinzugefügt: $replaceWith, $vectorSearch, $merge, $set, $unset, $bucket und 3 neue Aggregationsoperatoren $pow, $rand, $dateTrunc. • Unterstützung für eine neue Version des Textindex hinzugefügt: Der Textindex v2 in Amazon DocumentDB 8.0 führt zusätzliche Token ein und verbessert so die Textsuchfunktionen. • Beinhaltet Verbesserungen der Vektorsuche durch parallele Erstellung von Vektorindizes, wodurch die Indexerstellungszeit um das bis zu 30-fache reduziert wird. • Die Unterstützung für TLS 1.0 und TLS 1.1 wurde entfernt. Die Mindestversion von TLS ist jetzt TLS 1.2.

Amazon DocumentDB 5.0

Versionshinweise für die Amazon DocumentDB-Engine Version 5.0 (MongoDB 5.0-Kompatibilität).

Amazon DocumentDB 5.0.2

Date	Versionshinweise
28. September 2026	Amazon DocumentDB 5.0.2 (Maschinenversion: 3.0.23065) Neue Features und Verbesserungen 1. Unterstützung für versteckte Indizes für alle Indextypen hinzugefügt (Planer-Version 2 und höher). 2. Unterstützung für die Methoden `min()` und `max()` Cursor im Befehl find. 3. Unterstützung für Indexfilter für alle Befehle hinzugefügt. 4. Der Abfrageplaner priorisiert jetzt eindeutige Indizes für Equality () `$eq` -Prädikate. 5. Das Kürzungslimit für Aggregatbefehle in der Ausgabe wurde erhöht. `currentOp` Fehlerbehebungen 1. Ein Fehler im Abfrageplaner der Version 1 (Standard) wurde behoben. 2. Beim Major Version Upgrade (MVU) wurden Textumschläge für teilweise Indexausdrücke behoben. 3. Ein Profiler-Problem bei Abfragen mit schneller Zählung wurde behoben.

Amazon DocumentDB 5.0.1

Date	Versionshinweise
18. August 2026	Sicherheits- und Stabilitätsverbesserungen Amazon DocumentDB 5.0.1 (Engine-Version: 3.0.22319) [in Bearbeitung] Dieser Engine-Patch bietet die folgenden Verbesserungen: • Verbesserte Anforderungsverarbeitung Amazon DocumentDB validiert eingehende Client-Anfragen jetzt gründlicher entlang des Verbindungs- und Nachrichtenanalysepfads. Dies verbessert die Widerstandsfähigkeit und Stabilität bei einer breiteren Palette von Eingabemustern. • Härtung des Authentifizierungs-Handshakes Amazon DocumentDB validiert den Authentifizierungsaustausch jetzt gründlicher. Dies verbessert die Robustheit beim Umgang mit unerwarteten oder falsch formatierten Eingaben. Wenn Sie ihn nicht früher anwenden, wendet Amazon DocumentDB diesen Patch automatisch während Ihres nächsten Wartungsfensters 30 Tage nach der Veröffentlichung an.
11. August 2026	Neues Feature Amazon DocumentDB 5.0.1 (Maschinenversion: 3.0.21138) Unterstützung für Authentifizierung aktiviert. SCRAM-SHA-256
8. Juni 2026	Neue Nebenversion Amazon DocumentDB 5.0.1 (Motorversion 3.0.21138) Amazon DocumentDB 5.0.1 ist die erste Nebenversion für die Hauptversion von Amazon DocumentDB 5.0. Nebenversionen enthalten Verbesserungen wie neue Funktionen, Sicherheitsupdates und Bugfixes. Sie können mit einem Nebenversions-Upgrade von

Date	Versionshinweise
	5.0.0 auf 5.0.1 aktualisieren. Weitere Informationen finden Sie unter Upgrade der Nebenversion von Amazon DocumentDB. Diese Nebenversion enthält die folgenden neuen Funktionen und Verbesserungen: 1. Metriken auf Sammelebene wurden hinzugefügt `gcRuntimeStats` und nach `storageSegmentBase` und `storageSegmentExtended` nach aufgeteilt, `collstats` um mehr Informationen über den Gesamtzustand des Garbage Collection-Prozesses zu erhalten. `documentFragmentStats` Details dazu finden Sie unter Müllabfuhr in Amazon DocumentDB. 2. Unterstützung für Operatoren `$rand` und `$pow` Aggregationsoperatoren. 3. Sie können die `Connections (active)` Metrik jetzt für Instanzen überwachen. Weitere Informationen finden Sie unter Instanzkontingente. 4. Die regionsübergreifende Stabilität globaler Cluster wurde verbessert, da der sekundäre Cluster-Neustart mit Global Survivable Read Only entfällt. 5. Unterstützung für Operatoren `$dateToParts` und `$dateFromParts` Aggregationsoperatoren. 6. Command-level Leistungsmetriken sind jetzt CloudWatch für CRUD-Operationen (`find`,,,`insert`,`findAndModify` ,`update`,`delete`,`aggregate` ,`count`,`distinct`,`commitTransaction`) `getMoreabortTransaction` , `AvgDuration` Exposure und `MaxConcurrent` Dimensionen für eine verbesserte Betriebsüberwachung verfügbar. `P100Duration` 7. `$replaceRoot` gibt `newRoot` Felder aus, die boolesche Werte (`true`/`false`) und numerische Werte (/) enthalten. `0` `1` 8. `$indexOfArray` unterstützt eine breitere Palette gültiger Abfrageausdrücke.

Date	Versionshinweise
	9. Der Explain-Plan beinhaltet `docsExamined` Metriken in der `executionStats` Ausgabe. 10. Die Leistung der SASL-Authentifizierung wurde verbessert.

Amazon DocumentDB 5.0.0

Date	Versionshinweise
18. August 2026	Sicherheits- und Stabilitätsverbesserungen Amazon DocumentDB 5.0.0 (Engine-Version: 3.0.22303) [in Bearbeitung] Dieser Engine-Patch bietet die folgenden Verbesserungen: • Verbesserte Anforderungsverarbeitung Amazon DocumentDB validiert eingehende Client-Anfragen jetzt gründlicher entlang des Verbindungs- und Nachrichtenanalysepfads. Dies verbessert die Widerstandsfähigkeit und Stabilität bei einer breiteren Palette von Eingabemustern. • Härtung des Authentifizierungs-Handshakes Amazon DocumentDB validiert den Authentifizierungsaustausch jetzt gründlicher. Dies verbessert die Robustheit beim Umgang mit unerwarteten oder falsch formatierten Eingaben. • Access-control Durchsetzung Amazon DocumentDB überprüft jetzt konsistent die erforderlichen Rechte auf allen Verschachtelungsebenen für verschachtelte Abfragevorgänge. Wenn Sie ihn nicht früher anwenden, wendet Amazon DocumentDB diesen Patch automatisch während Ihres nächsten Wartungsfensters 30 Tage nach der Veröffentlichung an.
6. Mai 2026	Fehlerbehebungen und andere Änderungen

Date	Versionshinweise
	Amazon DocumentDB 5.0 (Engine-Version: 3.0.20421) Diese Version bezieht sich auf einen erforderlichen Sicherheitspatch.
20. April 2026	Neues Feature In-place Ein Hauptversions-Upgrade (MVU) von Amazon DocumentDB 5.0 auf Amazon DocumentDB 8.0 ist jetzt verfügbar. Weitere Informationen finden Sie unter Direktes Upgrade der Hauptversion von Amazon DocumentDB.
13. April 2026	Fehlerbehebungen und andere Änderungen Amazon DocumentDB 5.0 (Engine-Versionen: 3.0.19917) Dies ist ein erforderlicher Engine-Patch, der die folgenden neuen Funktionen, Verbesserungen und Korrekturen enthält: 1. Die `DatabaseConnectionsActiveMax` Metrik zeichnet jetzt `Connections (active)` nach Instanzen auf. Weitere Informationen finden Sie unter Instanzkontingente. 2. Die Leistung der SASL-Authentifizierung wurde verbessert. Alle Amazon DocumentDB 5.0-Cluster werden im nächsten Wartungsfenster automatisch auf diese Patch-Version aktualisiert.

Date	Versionshinweise
22. Oktober 2025	Neue Features Amazon DocumentDB 5.0 (Engine-Patch-Version 3.0.17983) Amazon DocumentDB unterstützt jetzt die neueste Generation speicheroptimierter Instances, db.r8g, die auf Arm-based AWS Graviton4-Prozessoren basieren und eine bis zu 30% bessere Leistung als R6G-Instances bieten. Weitere Informationen finden Sie in diesem Handbuch und in diesem Blogbeitrag. Instanzklassen verwalten https://aws.amazon.com/about-aws/whats-new/2025/10/amazon-documentdb-graviton4-based-r8g-database-instances/ Neue Features Amazon DocumentDB 5.0 (Engine-Patch-Version 3.0.15902) • Unterstützung für längere Sammlungsnamen hinzugefügt (Namespace-Länge bis zu 255 Zeichen). • Es wurde ein neuer Abfrageplaner eingeführt (Planer-Version 2.0). Weitere Informationen finden Sie unter Abfrageplaner v2 in diesem Handbuch. • Es wurden neue Felder `collstats` für „mvccidStats“ und „gcRuntimeStats“ hinzugefügt, die Informationen über den allgemeinen Zustand der Sammlung in Bezug auf Garbage Collection und alternde Sammlungen liefern.
16. Oktober 2025	Neues Feature Amazon DocumentDB 5.0 (Engine-Patch-Version 3.0.15902) Amazon DocumentDB unterstützt jetzt den Dual-Stack-Modus (IPv4/IPv6), wenn sich Ihr Cluster in einer Virtual Private Cloud (VPC) befindet. Weitere Informationen finden Sie in diesem Blogbeitrag und Amazon VPC und Amazon DocumentDB im Amazon DocumentDB Developer Guide.

Date	Versionshinweise
15. September 2025	Fehlerbehebungen und andere Änderungen Amazon DocumentDB 5.0 (Engine-Patch-Version 3.0.17983) Engine-Absturz behoben, wenn Abfragen nachfolgende UND-Stufen enthielten. `$replaceRoot $lookup`

Date	Versionshinweise
29. Juli 2025	Neue Features Amazon DocumentDB 5.0 (Engine-Patch-Version 3.0.15902) • Unterstützung für den Aggregationsoperator hinzugefügt. `$regexFindAll` • Index-Bloat- und sammlungsbasierte Bloat-Metriken wurden eingeführt • Es wurden CloudWatch `AvailableMVCCIds` Metriken hinzugefü gt`LongestActiveGCRuntime` , die Informationen über den allgemeinen Zustand des Clusters in Bezug auf Garbage Collection und veraltete Sammlungen liefern. • Der `serverStatus` Befehl enthält jetzt ein Feld „nvme_cache“, das die Anzahl der Seiten anzeigt, die auf Instances in den NVMe-Cache geschrieben und nicht geschrieben wurden. NVMe-backed • Die Beschränkung der Anzahl der Felder für und wurde entfernt `$group` `$project` • Unterstützung für die folgenden String-Operatoren: `$trim` `$rtrim`,`$ltrim`,`$regexFindAll` ,`$replaceOne` , und `$replaceAll` • Unterstützung für die folgenden Array-Operatoren: `$first` und `$last` • Unterstützung für den folgenden Stage-Operator: `$collStats` Fehlerbehebungen und andere Änderungen Amazon DocumentDB 5.0 (Engine-Patch-Version 3.0.15902) Es wurde ein Problem bei der Indexerstellung behoben, das große Sammlungen betraf.

Date	Versionshinweise
28. Juli 2025	Neue Features Amazon DocumentDB 5.0 (Engine-Patch, Version 3.0.14228) Auf Amazon DocumentDB-Instances basierende Cluster unterstützen jetzt Amazon DocumentDB Serverless, eine On-Demand-Konfigur ation mit automatischer Skalierung. Weitere Informationen finden Sie unter Serverlose Verwendung von Amazon DocumentDB und in diesem Blogbeitrag über Neuigkeiten.
24. März 2025	Neues Feature Amazon DocumentDB 5.0 (Engine-Patch-Version 3.0.11800) Unterstützung für `postBatchResumeToken` In-Change-Streams hinzugefügt. Weitere Informationen finden Sie unter Wiederaufnahme eines Change-Streams mit Post `BatchResumeToken` .
15. Januar 2025	Neue Features Amazon DocumentDB 5.0 (Engine-Patch-Version 3.0.11800) Amazon DocumentDB behält jetzt die Leseverfügbarkeit durch Neustarts der Writer-Instance bei. Reader-Instances bearbeiten nun weiterhin Leseanforderungen bei Neustarts der Writer-Instances. Fehlerbehebungen und andere Änderungen Amazon DocumentDB 5.0 (Engine-Patch-Version 3.0.11800) • Der `killOp` Befehl wurde behoben, um einen speziellen Fall von Masseneinfügungen zu behandeln. • Die Nutzung des I/O-related Netzwerkspeichers auf Amazon DocumentDB-Instances wurde verbessert. • `count`Befehl für `$text` Filterabfragen behoben.

Date	Versionshinweise
18. Dezember 2024	Neue Features Amazon DocumentDB 5.0 (Engine-Patch-Version 3.0.5721) Auf Amazon DocumentDB-Instances basierende Cluster unterstützen jetzt R6GD-Instances. NVMe-backed Weitere Informationen finden Sie unter NVMe-backed Instanzen und in diesem Blogbeitrag über Neuigkeiten.
6. November 2024	Neue Features Amazon DocumentDB 5.0 (Engine-Patch-Version 3.0.11051) • Unterstützung für TLS-Mindestversionen `tls1.2+` und `tls1.3+` Cluster-Parameter hinzugefügt. `tls` • Unterstützung für die Zeichen Dollar ($) und Punkt (.) in Feldnamen aktiviert. Informationen zu funktionalen Unterschieden finden Sie unterDollar ($) und Punkt (.) in Feldnamen.

Date	Versionshinweise
17. September 2024	Neue Features Amazon DocumentDB 5.0 (Engine-Patch-Version 3.0.10696) Amazon DocumentDB unterstützt jetzt das Neuerstellen von Indizes mit in der. `reIndex` `runCommand` Weitere Informationen finden Sie unter Indexverwaltung mithilfe von `Reindex`. **Note** `reIndex`wird nur auf Amazon DocumentDB 5.0 (Engine Patch Version 3.0.10696 und höher) unterstützt. Fehlerbehebungen und andere Änderungen Amazon DocumentDB 5.0 (Engine-Patch-Version 3.0.10696) • `$setOnInsert` unterstützt jetzt das `_id` Feld bei Einfügungen, die aus Aktualisierungsvorgängen resultieren. • Es wurde ein Problem behoben, das verhindert, dass Speicherplatz nach einem Upgrade der Hauptversion auf Amazon DocumentDB 5.0 zurückgewonnen wird.

Date	Versionshinweise
22. August 2024	Neues Feature Amazon DocumentDB 5.0 (Alle Engine-Patch-Versionen) Globale Amazon DocumentDB-Cluster unterstützen jetzt Cluster-Switchovers und verwaltete Cluster-Failover. Weitere Informationen erhalten Sie unter Durchführung eines Switchovers für einen globalen Amazon DocumentDB-Cluster und Durchführung eines verwalteten Failovers für einen globalen Amazon DocumentDB-Cluster. **Note** Globale Cluster-Switchover und -Failover werden nur in Amazon DocumentDB 4.0 und 5.0 unterstützt.

<table>
<tr><th>Date</th><th>Versionshinweise</th></tr>
<tr><td>23. Juli 2024</td><td>
Neue Features

Amazon DocumentDB 5.0 (Engine-Patch, Version 3.0.8126)

• Unterstützung für längere Indexnamen (bis zu 255 Zeichen) hinzugefügt. Weitere Informationen finden Sie unter Benennung seinschränkungen:.
• Das maximale Verbindungslimit wurde um das Zweifache erhöht:
<table>
<tr><th>Instance-Typ</th><th>Ursprüngliches Limit</th><th>Neues Limit</th></tr>
<tr><td>t3.medium</td><td>500</td><td>1000</td></tr>
<tr><td>t4g.medium</td><td>500</td><td>1000</td></tr>
<tr><td>r5.large</td><td>1700</td><td>3400</td></tr>
<tr><td>r5.xlarge</td><td>3500</td><td>7000</td></tr>
<tr><td>r5.2xlarge</td><td>7100</td><td>14200</td></tr>
<tr><td>r5.4xlarge</td><td>14200</td><td>28400</td></tr>
<tr><td>r5.8xlarge</td><td>28400</td><td>60000</td></tr>
<tr><td>r5.12xlarge</td><td>30000</td><td>60000</td></tr>
<tr><td>r5.16xlarge</td><td>30000</td><td>60000</td></tr>
<tr><td>r5.24xlarge</td><td>30000</td><td>60000</td></tr>
<tr><td>r6g.large</td><td>1700</td><td>3400</td></tr>
<tr><td>r6g.xlarge</td><td>3500</td><td>7000</td></tr>
<tr><td>r6g.2xlarge</td><td>7100</td><td>14200</td></tr>
</table>
</td></tr>
</table>

Date	Versionshinweise

Instance-Typ	Ursprüngliches Limit	Neues Limit
r6g.4xlarge	14200	28400
r6g.8xlarge	28400	60000
r6g.12xlarge	30000	60000
r6g.16xlarge	30000	60000

Weitere Informationen finden Sie unter Kontingente für Instanzen.

Fehlerbehebungen und andere Änderungen

Amazon DocumentDB 5.0 (Engine-Patch-Version 3.0.8126)

Die Logik für die Synchronisation `CurrentTime` und `ResumeToken` für den Change-Stream auf den Lesern wurde verbessert.

Date	Versionshinweise

22. Juli 2024	Neue Features
	
	Amazon DocumentDB 5.0 (Engine-Patch, Version 3.0.6742)
	
	• Unterstützung für DML Audit-Filterung hinzugefügt. Sie können jetzt Filterbedingungen einrichten, um DML-Audit-Logs basierend auf ihren spezifischen Anforderungen herauszufiltern, anstatt jede DML-Abfrage zu protokollieren. Weitere Informationen finden Sie unter Filterung von DML-Audit-Ereignissen.
	• Unterstützung bei der Dokumentenkomprimierung für Folgendes hinzugefügt:
	  • Festlegung eines Mindestschwellenwerts für die Komprimierung
	  • Komprimierung für bestehende Sammlungen aktivieren (gilt für neue Dokumente)
	  • Erlauben Sie die Standardkomprimierungseinstellung auf Clusterebene
	
	  Weitere Informationen finden Sie unter Verwaltung der Dokumentenkomprimierung auf Sammlungsebene.
	• Unterstützung für die Nutzung von Change-Streams auf Reader-Instances wurde hinzugefügt. Weitere Informationen finden Sie unter Verwendung von Change Streams auf sekundären Instances.

9. Juli 2024	Neues Feature Für Amazon DocumentDB 5.0 (Engine-Patch-Version 3.0.4780) ist für TLS-CA-Zertifikatsaktualisierungen kein Systemneustart mehr erforderlich. Weitere Informationen finden Sie unter Aktualisierung Ihrer Amazon DocumentDB-TLS-Zertifikate.

Date	Versionshinweise
29. Mai 2024	Neue Features Amazon DocumentDB 5.0 (Engine-Patch, Version 3.0.6742) • Unterstützung für und -Operatoren hinzugefügt. `$regexMatch` `$regexFind` • Unterstützung hinzugefügt, um bei der Bearbeitung großer Ganzzahlen die volle Genauigkeit der Auditprotokolle zu gewährlei sten. In den Auditprotokollen wird jetzt die exakte numerisch e Darstellung aller Zahlen beibehalten, sodass ein Verlust an Genauigkeit vermieden wird.
3. April 2024	Neue Features Amazon DocumentDB 5.0 (Engine-Patch-Version 3.0.5721) • Unterstützung `bypassDocumentValidation` und detaillie rte Fehlermeldung für hinzugefügt. `$jsonSchema` Mehr über `bypassDocumentValidation` erfahren Sie unter umgehen DocumentValidation. • Unterstützung von `$expr` hinzugefügt. • Unterstützung für unkorrelierte Joins in hinzugefügt. `$lookup` • Unterstützung hinzugefügt, um Validierungsregeln in der `$out` Aggregationsphase beizubehalten.

Date	Versionshinweise
10. Januar 2024	Neue Features Amazon DocumentDB 5.0 (Engine-Patch-Versionen 3.0.4574, 3.0.4780, 3.0.4960) • Unterstützung für HNSW-Vektorindizes hinzugefügt. Weitere Informationen finden Sie unter Vektorsuche für Amazon DocumentDB. • Unterstützung für Teilindizes hinzugefügt. Weitere Informationen finden Sie unter Teilweiser Index. • Es wurde eine Unterstützung für die GC-Laufzeit für eine Sammlung innerhalb des `currentOp` Befehls hinzugefügt. • Unterstützung für Textindexe für die native Textsuche in Amazon DocumentDB hinzugefügt. Weitere Informationen finden Sie unter Textsuche mit Amazon DocumentDB durchführen. • Unterstützung für `$jsonSchema` Schema-Schlüsselwörtertype,`allOf`,`oneOf`,`anyOf`,`not`,`maxItems`,`minItems`,`maxProperties` ,`minProperties` ,`pattern`, `patternProperties` `multipleOf` `dependencies` , und `uniqueItems` hinzugefügt. Weitere Informationen finden Sie unter Verwenden der JSON-Schemavalidierung. • Unterstützung für arithmetische Operatoren`$ceil`,,`$floor`, `$ln` `$log` `$log10` `$sqrt`, und hinzugefügt. `$exp` Weitere Informationen finden Sie unter Arithmetische Operatoren. • Unterstützung für den Operator für bedingte Ausdrücke hinzugefügt. `$switch` • Unterstützung für parallele `IVFFLAT` Vektor-Index-Builds hinzugefügt. Die Dokumentation wurde aktualisiert, indem die Beschränkung für parallele `IVFFLAT` Vektorindex-Builds aus dem Entwicklerhandbuch entfernt wurde. Fehlerbehebungen und andere Änderungen

Date	Versionshinweise
	• Beim Aufrufen wurde eine Funktion hinzugefügt, bei der Groß- und Kleinschreibung nicht berücksichtigt wird. `db.runCommand("dbstats")` Amazon DocumentDB 5.0-Kunden, die Engine-Patch-Versionen vor 3.0.4960 oder 2.0.10221 verwenden, sollten diese neuesten Engine-Patches installieren. • Ein Fehler beim Aufrufen `db.coll.stats()` auf der Mongo-Shell-Version 1.7 und höher wurde behoben. Die Dokumentation wurde aktualisiert, indem der Tipp `db.coll.stats()` zur Fehlerbehebung bei der Mongo-Shell aus dem Entwicklerhandbuch entfernt wurde.
29. November 2023	Neue Features Amazon DocumentDB 5.0 (Engine-Patch-Version 3.0.3727) Unterstützung für Vektorsuche hinzugefügt. Weitere Informationen finden Sie in diesem Blogbeitrag und Vektorsuche für Amazon DocumentDB im Amazon DocumentDB Developer Guide.
21. November 2023	Neue Features Amazon DocumentDB 5.0 (Engine-Patch-Version 3.0.3727) Unterstützung für Speicher hinzugefügt. I/O-optimized Weitere Informationen finden Sie Amazon DocumentDB-Cluster-Speicherkonfigurationen im Amazon DocumentDB Developer Guide. Integration für maschinelles Lernen ohne Code mit SageMaker Canvas hinzugefügt. Weitere Informationen finden Sie No-code maschinelles Lernen mit Amazon SageMaker AI Canvas im Amazon DocumentDB Developer Guide.

Date	Versionshinweise
6. November 2023	Amazon DocumentDB 5.0 (Engine-Patch-Versionen 3.0.3727) Neue Features • Unterstützung für `$jsonSchema` Schema-Schlüsselwörtern `maxLength`, `minLength`, `maximum`, `minimum`, `exclusiveMaximum`, `exclusiveMinimum` und `items` hinzugefügt. `additionalItems` Beachten Sie, dass die JSON-Schemavalidierung nur in instanzbasierten Clustern unterstützt wird. • Unterstützung für den `$convert` Aggregationspipeline-Operator und die daraus abgeleiteten Kurzoperatoren `$toBool`, `$toInt`, `$toLong`, `$toDouble`, `$toString` und wurde hinzugefügt. `$toDecimal` `$toObjectId` `$toDate` • Unterstützung für die Operatoren „Set Expression“ `$setDifference`, und `$anyElementTrue` hinzugefügt. `$allElementTrue` Fehlerbehebungen und andere Änderungen Es wurde ein Problem behoben, bei dem ein Change-Stream-Update von `-NaN` bis nicht angezeigt `NaN` wurde.
15. September 2023	Neue Features Amazon DocumentDB 5.0 (Engine-Patch, Version 3.0.3140) Unterstützung für den $JSONSchema-Schema-Validator wurde nur in instanzbasierten Clustern hinzugefügt. Weitere Informationen finden Sie unter Verwenden der JSON-Schemavalidierung.

Date	Versionshinweise
13. Juli 2023	Neue Features Amazon DocumentDB 5.0 (Engine-Patch-Version 3.0.1948) • Unterstützung für die Komprimierung von Dokumenten hinzugefügt. • Unterstützung für parallele Indexerstellungen hinzugefügt. • Unterstützung für den Status der Indexerstellung hinzugefügt. Fehlerbehebungen und andere Änderungen Amazon DocumentDB 5.0 (Engine-Patch-Version 3.0.1948) • Das Authentifizierungsproblem bei `createCollection` elastischen Amazon DocumentDB-Clustern, bei dem Benutzer keinen Zugriff auf Systemsammlungen hatten, wurde behoben. • Es wurde ein Problem behoben, bei dem sekundäre Region-Instances nicht dieselben Instance-Namen der primären Region verwenden konnten.
7. Juni 2023	Fehlerbehebungen und andere Änderungen Amazon DocumentDB 5.0 • r5- und t3.medium-Instances werden jetzt in Amazon DocumentDB 5.0 unterstützt. • `engineVersion` Die Standardoption ist `5.0.0` im AWS SDK, und. AWS CLI CloudFormation

Date	Versionshinweise
10. Mai 2023	Fehlerbehebungen und andere Änderungen Amazon DocumentDB 5.0 (Engine-Patch-Version 3.0.1361) • Unterstützung für `ignoreunknownindexoptions` in dem Befehl hinzugefügt. createIndex • Das Hinzufügen interner Überwachungsabfragen zu den Überwachungsprotokollen wurde beendet. • Benutzerdefinierte lokale Variablennamen unterstützen jetzt „_“ (Unterstrich) für Projektionsoperatoren wie `$let` und`$filter`.

Date	Versionshinweise

1. März 2023	Neue Features
	
	Amazon DocumentDB 5.0 (Engine-Patch, Version 3.0.775)
	
	• Amazon DocumentDB 5.0 eingeführt
	  • MongoDB 5.0-Kompatibilität (Unterstützung für MongoDB 5.0-API-Treiber)
	  • Unterstützung für Verschlüsselung auf Client-side Feldebene (FLE). Sie können jetzt Felder auf der Clientseite verschlüs seln, bevor Sie die Daten in den Amazon DocumentDB-Cluster schreiben. Weitere Informationen finden Sie unter Verschlüs selung auf Feldebene Client-side
	  • Neue Aggregationsoperatoren:`$dateAdd`, `$dateSubtract`
	• Das Speicherlimit wurde für alle instanzbasierten Amazon DocumentDB-Cluster und shard-basierten elastischen Cluster auf 128 TiB erhöht.
	• Amazon DocumentDB 5.0 unterstützt jetzt den Index-Scan mit dem Operator in der `$elemMatch` ersten Verschachtelungseb ene. Indexscans werden unterstützt, wenn Abfragen nur über eine `$elemMatch` Filterebene verfügen und die verschachtelte `$elemMatch` Abfrage den Indexscan nicht unterstützt.
	
	Abfrage-Shape, das Index-Scan unterstützt:
	
	`db.foo.find( { "a": {$elemMatch: { "b": "xyz", "c": "abc"} } })`
	
	Abfrage-Shape, das den Indexscan nicht unterstützt:
	
	`db.foo.find( { "a": {$elemMatch: { "b": {$elemMatch: { "d": "xyz", "e": "abc"} }} } })`

Amazon DocumentDB 4.0

Versionshinweise für die Amazon DocumentDB-Engine Version 4.0 (MongoDB 4.0-Kompatibilität).

Date	Versionshinweise
31. August 2026	Neues Feature Amazon DocumentDB unterstützt jetzt direkte Hauptversions-Upgrades (MVU) direkt von Engine-Version 4.0 auf Version 8.0. Dadurch sind keine Zwischenversions-Upgrades erforderlich und Sie können Ihre 4.0-Cluster aktualisieren und gleichzeitig vorhanden e Daten, Konfigurationen und Cluster-Einstellungen beibehalten. Dieses Upgrade ist in allen AWS Regionen verfügbar, in denen diese Versionen derzeit unterstützt werden. Weitere Informationen finden Sie unter Direktes Upgrade der Hauptversion von Amazon DocumentD B.
18. August 2026	Sicherheits- und Stabilitätsverbesserungen Amazon DocumentDB 4.0 (Engine-Version: 2.0.14287) [in Bearbeitu ng] Dieser Engine-Patch bietet die folgenden Verbesserungen: • Verbesserte Anforderungsverarbeitung Amazon DocumentDB validiert eingehende Client-Anfragen jetzt gründlicher entlang des Verbindungs- und Nachrichtenanalysepfads. Dies verbessert die Widerstandsfähigkeit und Stabilität bei einer breiteren Palette von Eingabemustern. • Härtung des Authentifizierungs-Handshakes Amazon DocumentD B validiert den Authentifizierungsaustausch jetzt gründlicher. Dies verbessert die Robustheit beim Umgang mit unerwarteten oder falsch formatierten Eingaben. • Access-control Durchsetzung Amazon DocumentDB überprüft jetzt konsistent die erforderlichen Rechte auf allen Verschachtelungseb enen für verschachtelte Abfragevorgänge.

Date	Versionshinweise
	Wenn Sie ihn nicht früher anwenden, wendet Amazon DocumentDB diesen Patch automatisch während Ihres nächsten Wartungsfensters 30 Tage nach der Veröffentlichung an.
27. April 2026	Fehlerbehebungen und andere Änderungen Amazon DocumentDB 4.0 (Engine-Version: 2.0.14278) Dies ist ein erforderlicher Engine-Patch, der die folgenden Probleme behebt: 1. Es wurde ein Problem behoben, das in seltenen Fällen dazu führen konnte, dass Point-in-Time Recovery (PITR) und Hauptversions-Upgrades (MVU) nicht mehr reagierten. Alle Amazon DocumentDB 4.0-Cluster werden im nächsten Wartungsfenster automatisch auf diese Patch-Version aktualisiert.
9. März 2026	Fehlerbehebungen und andere Änderungen Amazon DocumentDB 4.0 (Engine-Version: 2.0.14165) Dieses Update bietet einen optionalen Engine-Patch mit den folgenden Verbesserungen: 1. Die `DatabaseConnectionsActiveMax` Metrik verfolgt jetzt `Connections (active)` nach Instanzen. Weitere Informationen finden Sie unter Instanzkontingente. 2. Die Leistung der SASL-Authentifizierung wurde verbessert.

Date	Versionshinweise
22. Oktober 2025	Neue Features Amazon DocumentDB 4.0 (Engine-Patch-Version 2.0.11747) Es wurden neue Felder `collstats` für „mvccidStats“ und „gcRuntimeStats“ hinzugefügt, die Informationen über den allgemeinen Zustand der Sammlung in Bezug auf Garbage Collection und alternde Sammlungen bereitstellen.
16. Oktober 2025	Neues Feature Amazon DocumentDB 4.0 (Engine-Patch-Version 2.0.11747) Amazon DocumentDB unterstützt jetzt den Dual-Stack-Modus (IPv4/IPv6), wenn sich Ihr Cluster in einer Virtual Private Cloud (VPC) befindet. Weitere Informationen finden Sie in diesem Blogbeitrag und Amazon VPC und Amazon DocumentDB im Amazon DocumentDB Developer Guide.

Date	Versionshinweise
29. Juli 2025	Neue Features Amazon DocumentDB 4.0 (Engine-Patch-Version 2.0.11747) • Index-Bloat- und sammlungsbasierte Bloat-Metriken wurden eingeführt • Es wurden CloudWatch `AvailableMVCCIds` Metriken hinzugefügt `LongestActiveGCRuntime`, die Informationen über den allgemeinen Zustand des Clusters in Bezug auf Garbage Collection und veraltete Sammlungen liefern. • Unterstützung für die folgenden String-Operatoren: `$trim` `$rtrim`, und `$ltrim` • Unterstützung für den folgenden Stage-Operator: `$collStats` Fehlerbehebungen und andere Änderungen Amazon DocumentDB 4.0 (Engine-Patch-Version 2.0.11747) Es wurde ein Problem bei der Indexerstellung behoben, das große Sammlungen betraf.
15. Januar 2025	Fehlerbehebungen und andere Änderungen Amazon DocumentDB 4.0 (Engine-Patch-Version 2.0.11096) • Der `killOp` Befehl zur Behandlung eines Sonderfalls von Masseneinträgen wurde behoben. • Die Nutzung des I/O-related Netzwerkspeichers auf Amazon DocumentDB-Instances wurde verbessert.
6. November 2024	Neue Features Amazon DocumentDB 4.0 (Engine-Patch-Version 2.0.10980) • Unterstützung für TLS-Mindestversionen `tls1.2+` und Cluster-Parameter `tls1.3+` hinzugefügt `tls`.

Date	Versionshinweise
17. September 2024	Fehlerbehebungen und andere Änderungen Amazon DocumentDB 4.0 (Engine-Patch-Version 2.0.10898) • `$setOnInsert` unterstützt jetzt das `_id` Feld bei Einfügungen, die aus Aktualisierungsvorgängen resultieren. • Es wurde ein Problem behoben, das verhindert, dass Speicherplatz nach einem Upgrade der Hauptversion auf Amazon DocumentDB 5.0 zurückgewonnen wird.
22. August 2024	Neues Feature Amazon DocumentDB 4.0 (Engine-Patch, Version 2.0.5704) Globale Amazon DocumentDB-Cluster unterstützen jetzt Cluster-Switchover und verwaltete Cluster-Failover. Weitere Informationen erhalten Sie unter Durchführung eines Switchovers für einen globalen Amazon DocumentDB-Cluster und Durchführung eines verwalteten Failovers für einen globalen Amazon DocumentDB-Cluster. **Note** Globale Cluster-Switchover und -Failover werden nur in Amazon DocumentDB 4.0 und 5.0 unterstützt.

<table>
<tr><th>Date</th><th>Versionshinweise</th></tr>
<tr><td>23. Juli 2024</td><td>
Neue Features

Amazon DocumentDB 4.0 (Engine-Patch-Version 2.0.10709)

• Unterstützung für längere Indexnamen (bis zu 255 Zeichen) hinzugefügt. Weitere Informationen finden Sie unter Benennung seinschränkungen:.
• Das maximale Verbindungslimit wurde um das Zweifache erhöht:
<table>
<tr><th>Instance-Typ</th><th>Ursprüngliches Limit</th><th>Neues Limit</th></tr>
<tr><td>t3.medium</td><td>500</td><td>1000</td></tr>
<tr><td>t4g.medium</td><td>500</td><td>1000</td></tr>
<tr><td>r5.large</td><td>1700</td><td>3400</td></tr>
<tr><td>r5.xlarge</td><td>3500</td><td>7000</td></tr>
<tr><td>r5.2xlarge</td><td>7100</td><td>14200</td></tr>
<tr><td>r5.4xlarge</td><td>14200</td><td>28400</td></tr>
<tr><td>r5.8xlarge</td><td>28400</td><td>60000</td></tr>
<tr><td>r5.12xlarge</td><td>30000</td><td>60000</td></tr>
<tr><td>r5.16xlarge</td><td>30000</td><td>60000</td></tr>
<tr><td>r5.24xlarge</td><td>30000</td><td>60000</td></tr>
<tr><td>r6g.large</td><td>1700</td><td>3400</td></tr>
<tr><td>r6g.xlarge</td><td>3500</td><td>7000</td></tr>
<tr><td>r6g.2xlarge</td><td>7100</td><td>14200</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Date</th><th>Versionshinweise</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance-Typ</th><th>Ursprüngliches Limit</th><th>Neues Limit</th></tr>
<tr><td>r6g.4xlarge</td><td>14200</td><td>28400</td></tr>
<tr><td>r6g.8xlarge</td><td>28400</td><td>60000</td></tr>
<tr><td>r6g.12xlarge</td><td>30000</td><td>60000</td></tr>
<tr><td>r6g.16xlarge</td><td>30000</td><td>60000</td></tr>
</table>
Weitere Informationen finden Sie unter Kontingente für Instanzen.
</td></tr>
<tr><td>22. Juli 2024</td><td>Neue Features

Amazon DocumentDB 4.0 (Engine-Patch-Version 2.0.10593)

• Unterstützung für DML-Audit-Filterung hinzugefügt. Sie können jetzt Filterbedingungen einrichten, um DML-Audit-Logs basierend auf ihren spezifischen Anforderungen herauszufiltern, anstatt jede DML-Abfrage zu protokollieren. Weitere Informationen finden Sie unter Filterung von DML-Audit-Ereignissen.</td></tr>
<tr><td>9. Juli 2024</td><td>Neues Feature

Für Amazon DocumentDB 4.0 (Engine-Patch-Version 2.0.10179) ist für TLS-CA-Zertifikatsaktualisierungen kein Systemneustart mehr erforderlich. Weitere Informationen finden Sie unter Aktualisierung Ihrer Amazon DocumentDB-TLS-Zertifikate.</td></tr>
</table>

Date	Versionshinweise
29. Mai 2024	Neue Features Amazon DocumentDB 4.0 (Engine-Patch-Version 2.0.10593) • Unterstützung hinzugefügt, um bei der Behandlung großer Ganzzahlen die volle Präzision in Audit-Logs zu gewährleisten. In den Auditprotokollen wird jetzt die exakte numerische Darstellung aller Zahlen beibehalten, sodass ein Verlust an Genauigkeit vermieden wird.
3. April 2024	Neue Features Amazon DocumentDB 4.0 (Engine-Patch-Version 2.0.10392) • Unterstützung für hinzugefügt. `bypassDocumentValidation` `$jsonSchema` Mehr über `bypassDocumentValidation` erfahren Sie unter umgehen DocumentValidation. • Unterstützung von hinzugefügt`$expr`. • Unterstützung für unkorrelierte Joins in hinzugefügt. `$lookup` • Unterstützung hinzugefügt, um Validierungsregeln in der `$out` Aggregationsphase beizubehalten.

Date	Versionshinweise
10. Januar 2024	Neue Features Amazon DocumentDB 4.0 (Engine-Patch-Versionen 2.0.10124, 2.0.10179, 2.0.10221) • Eine Unterstützung für die GC-Laufzeit wurde für eine Sammlung innerhalb des Befehls hinzugefügt. `current0p` • Unterstützung für `$jsonSchema` Schema-Schlüsselwörter `typeall0f`,`one0f`,`any0f`,`not`,`maxItems`,`minItems`,`maxProperties` , `minProperties pattern`,`patternProperties` , `multiple0f dependencies` , und hinzugefügt`uniqueItems` . Weitere Informationen finden Sie unter Verwenden der JSON-Schemavalidierung. • Unterstützung für arithmetische Operatoren`$ceil`,,`$floor`, `$ln` `$log` `$log10``$sqrt`, und hinzugefügt. `$exp`  Weitere Informationen finden Sie unter Arithmetische Operatoren. • Unterstützung für den Operator für bedingte Ausdrücke hinzugefügt. `$switch`
20. Dezember 2023	Weitere Änderungen Die Unterstützung für ein direktes Upgrade der Hauptversion in Amazon DocumentDB 4.0 wurde aktiviert. Weitere Informationen finden Sie unter Direktes Upgrade der Hauptversion von Amazon DocumentDB.

Date	Versionshinweise
6. November 2023	Amazon DocumentDB 4.0 (Engine-Patch-Versionen 2.0.9876) Neue Features • Unterstützung für `$jsonSchema` Schema-SchlüsselwörtermaxLength ,,minLength ,maximum, minimum exclusiveMaximum exclusiveMinimum , items und hinzugefügt. additionalItems Beachten Sie, dass die JSON-Schemavalidierung nur in instanzbasierten Clustern unterstützt wird. • Unterstützung für den `$convert` Aggregationspipeline-Operator und die daraus abgeleiteten Kurzoperatoren$toBool,,, $toInt $toLong$toDouble , $toString und wurde hinzugefügt. $toDecimal $toObjectId $toDate • Unterstützung für die Operatoren „Set Expression“$setDifference , und $anyElementTrue hinzugefügt. $allElementTrue Fehlerbehebungen und andere Änderungen Es wurde ein Problem behoben, bei dem ein Change-Stream-Update von -NaN bis nicht angezeigt NaN wurde.
20. September 2023	Neue Features Unterstützung für direkte Hauptversions-Upgrades in Amazon DocumentDB 4.0 wurde hinzugefügt. Weitere Informationen finden Sie unter Direktes Upgrade der Hauptversion von Amazon DocumentDB.

Date	Versionshinweise
15. September 2023	Neue Features Amazon DocumentDB 4.0 (Engine-Patch-Version 2.0.9686) Unterstützung für den $JSONSchema-Schema-Validator wurde nur in instanzbasierten Clustern hinzugefügt. Weitere Informationen finden Sie unter Verwenden der JSON-Schemavalidierung.
13. Juli 2023	Neue Features Amazon DocumentDB 4.0 (Engine-Patch-Version 2.0.9259) • Unterstützung für parallele Indexerstellungen hinzugefügt. Fehlerbehebungen und andere Änderungen Amazon DocumentDB 4.0 (Engine-Patch-Version 2.0.9259) • Das Hinzufügen interner Überwachungsabfragen zu den Audit-Pro tokollen wurde beendet.

Date	Versionshinweise
4. April 2023	Fehlerbehebungen und andere Änderungen Amazon DocumentDB 4.0 (Engine-Patch-Version 2.0.8934) • Es wurde ein Problem mit der DML-Überwachung behoben, wenn sie während einer laufenden Arbeitslast aktiviert wurde. • Problem mit der DML-Überwachung behoben, bei dem Aggregatb efehle mit Hinweis ein Zeichenfolgenwert übergeben wurden. • Es wurde ein Problem behoben, bei dem der `listCollections` Befehl nicht funktionierte, wenn Benutzer mit der Rolle readwrite anydatabase sowohl die Optionen authorizedCollections als auch NameOnly auf true gesetzt hatten. • Es gab ein Problem beim korrekten Analysieren einer numerischen Zeichenfolge in einem Feldnamen. Dieser Fehler wurde behoben. • Brechen Sie lang laufende Cursor ab, wenn sie sich auf die Speicherbereinigung auswirken. • Benutzerdefinierte lokale Variablennamen unterstützen jetzt „_“ (Unterstrich) für Projektionsoperatoren wie und. `$let` `$filter`
27. Februar 2023	Fehlerbehebungen und andere Änderungen Amazon DocumentDB 4.0 Unterstützung für AWS Lambda hinzugefügt. Weitere Informationen finden Sie unter Verwenden AWS Lambda mit Change Streams.

Date	Versionshinweise
09. August 2022	Neue Features 4.0 • Unterstützung für den Datentyp Decimal128 hinzugefügt. Der Decimal128 ist ein BSON-Datentyp, der in allen Regionen unterstützt wird, in denen DocumentDB verfügbar ist. Weitere Informationen finden Sie unter Data Types. • Unterstützung für die Prüfung von DML-Abfragen mit Amazon Logs wurde hinzugefügt. CloudWatch Jetzt kann Amazon DocumentDB DML-Ereignisse (Data Manipulation Language) und DDL-Ereignisse (Data Definition Language) in Amazon Logs aufzeichnen. CloudWatch Weitere Informationen finden Sie in diesem Blog-Beitrag. Fehlerbehebungen und andere Änderungen 4.0 • Sie können jetzt Ihr eigenes Passwort mit einem eigenen Passwort mit Privilegien ändern. `changeOwnPassword`
25. Juli 2022	Neue Features Amazon DocumentDB 4.0 Sie können Cluster jetzt schneller erstellen, da Sie Klone erstellen können, die dasselbe DocumentDB-Cluster-Volume verwenden und dieselben Daten wie der ursprüngliche Cluster haben. Einzelheiten finden Sie unter Amazon DocumentDB-Cluster verwalten.

Date	Versionshinweise
27. Juni 2022	Neue Features Amazon DocumentDB 4.0 (Engine-Patch-Version 2.0.7509) Amazon DocumentDB passt die Größe Ihrer Datenbank dynamisch auf der Grundlage von Nutzungsmustern an. Durch das Hinzufüge n weiterer Daten wird der Speicherplatz auf bis zu 64 Tebibyte (TiB) erhöht, und durch das Löschen von Daten wird der zugewiesene Speicherplatz verringert.
7. April 2022	Neue Features 4.0 (Engine-Patch-Versionen 2.0.6142 und 2.0.6948) Amazon DocumentDB Performance Insights befindet sich jetzt in der Vorschauversion. Sie können jetzt den Leistungsverlauf von sieben Tagen ohne zusätzliche Kosten in einem fortlaufenden Fenster speichern. Weitere Informationen finden Sie unter Überwachung mit Performance Insights.

Date	Versionshinweise

24. Januar 2022	Neue Features
	
	4.0 (Engine-Patch-Version 2.0.5170)
	
	• Amazon DocumentDB bietet jetzt eine kostenlose Testversion an. Einzelheiten finden Sie auf der Seite mit der kostenlosen Testversi on von Amazon DocumentDB.
	• Sie können jetzt erweiterte Funktionen mit Geospatial-Abfragen verwenden, einschließlich der folgenden APIs:
	  • `$geoWithin`
	  • `$geoIntersects`
	• Unterstützung für die folgenden MongoDB-Operatoren wurde hinzugefügt:
	  • `$mergeObjects`
	  • `$reduce`
	
	Weitere Informationen finden Sie unter Abfragen von Geodaten mit Amazon DocumentDB.

Date	Versionshinweise

21. Januar 2022	Neue Features
	
	Amazon DocumentDB 4.0 (Engine-Patch, Version 2.0.5706)
	
	• Amazon DocumentDB Graviton2-Instances (r6g.large, r6g.2xlarge, r6g.4xlarge, r6g.8xlarge, r6g.12xlarge, r6g.16xlarge und t4g.medium) werden jetzt unterstützt.
	
	Neue Features
	
	4.0 (Engine-Patch, Version 2.0.5706)
	
	• Unterstützung für die folgenden MongoDB-APIs wurde hinzugefügt:
	  • `$reduce`
	  • `$mergeObjects`
	  • `$geoWithin`
	  • `$geoIntersects`

Date	Versionshinweise

25. Oktober 2021	Neue Features
	
	4.0 (Engine-Patch-Version 2.0.5704)
	
	• Unterstützung für die folgenden MongoDB-APIs wurde hinzugefügt
	  • `$literal`
	  • `$map`
	  • `$$ROOT`
	• Unterstützung für GeoSpatial Abfragefunktionen. Weitere Informati onen finden Sie in diesem Blogbeitrag
	• Unterstützung für die Zugriffskontrolle mit benutzerdefinierten Rollen. Weitere Informationen finden Sie in diesem Blogbeitrag
	• Amazon DocumentDB JDBC-Treiber, um die Konnektivität von BI-Tools wie Tableau und Abfragetools wie SQL Workbench zu ermöglichen
	
	Fehlerbehebungen und andere Änderungen
	
	4.0 (Engine-Patch-Version 2.0.5704)
	
	• Bugfix für die korrekte Sortierung`$natural`, wenn ein explizites Zeichen zusammen `.sort()` mit `$natural`
	• Bugfix für Change Stream to work with `$redact`
	• Bugfix für `$ifNull` die Arbeit mit leerem Array
	• Bugfix für übermäßigen consumption/server Ressourcenabsturz, wenn ein aktuell angemeldeter Benutzer gelöscht wurde oder diesem Benutzer die Berechtigung für eine laufende Aktivität entzogen wurde
	• Bugfix und Rechteüberprüfung `listDatabase listColle ction`
	• Fehler behoben: Deduplizierungslogik für Elemente mit mehreren Schlüsseln

Date	Versionshinweise
24. Juni 2021	Neue Features 4.0 (Engine-Patch-Version 2.0.3371) • r5.8xlarge- und r5.16xlarge Instances werden jetzt unterstützt. Weitere Informationen finden Sie im Blogbeitrag Amazon DocumentDB unterstützt jetzt die Instances r5.8xlarge und r5.16xlarge. • Globale Cluster werden jetzt unterstützt, um Disaster Recovery nach Region-wide Ausfällen zu ermöglichen und globale Lesevorgänge mit niedriger Latenz zu ermöglichen, indem Lesevorgänge aus dem nächstgelegenen Amazon DocumentDB-Cluster zugelassen werden. Beachten Sie, dass globale Cluster derzeit in den Regionen Südamerika (São Paulo), Europa (Mailand), China (Peking) und China (Ningxia) nicht unterstützt werden.

Date	Versionshinweise
4. Mai 2021	Neue Features 4.0 (Engine-Patch-Version 2.0.3371) • `renameCollection` • `$zip` • `$indexOfArray` • `$reverseArray` • `$natural` • `$hint`Unterstützung für das Update • Index-Scan für `distinct` Fehlerbehebungen und andere Änderungen 4.0 (Engine-Patch-Version 2.0.3371) • Reduzierter Speicherverbrauch für Abfragen `$in` • Ein Speicherverlust in Multikey-Indizes wurde behoben • Der Erklärungsplan und die Profiler-Ausgabe für wurden korrigiert `$out` • Ein Timeout für Operationen aus dem internen Überwachu ngssystem wurde hinzugefügt, um die Zuverlässigkeit zu verbessern • Es wurde ein Fehler behoben, der sich auf die Abfrageprädikate auswirkte, die an Multikey-Indizes übergeben wurden
15. Januar 2021	Neue Features Amazon DocumentDB 4.0 (Engine-Patch-Version 2.0.722) • Keine

Date	Versionshinweise
9. November 2020	Neue Features Amazon DocumentDB 4.0 (Engine-Patch-Version 2.0.722) • MongoDB 4.0-Kompatibilität • ACID-Transaktionen • Unterstützung für `cluster(client.watch()` oder `mongo.watch())` und `(db.watch())` Change-Streams auf Datenbankebene • Möglichkeit, Change-Streams zu starten oder fortzusetzen mit `startAtOperationTime` • Verlängern Sie die Aufbewahrungsdauer Ihres Change-Streams auf 7 Tage (zuvor 24 Stunden) • AWS DMS Ziel für Amazon DocumentDB 4.0 • CloudWatch Metriken: `TransactionsOpen` `TransactionsOpenMax` , `TransactionsAborted` `TransactionsStarted` , und `TransactionsCommitted` • Neue Felder für Transaktionen in `currentOp` `ServerStatus` , und`profiler`. • Fähigkeit, einen Index in der `$lookup` Aggregationsphase zu verwenden • `find()`Abfragen mit Projektionen können direkt von einem Index aus bedient werden (abgedeckte Abfrage) • Fähigkeit zur Verwendung `hint()` mit dem `findAndModify` • Leistungsoptimierungen für `$addToSet` den Bediener • Verbesserungen zur Reduzierung der Gesamtindexgrößen. • Neue Aggregationsoperatoren: `$ifNull` `$replaceRoot` ,`$setIsSubset` , `$setInstersection` `$setUnion` , und `$setEquals` • Mit den `ListDatabase` Befehlen `ListCollection` und können Sie nun optional die `authorizedDatabases` Parameter `authorizedCollections` und verwenden, damit Benutzer

Date	Versionshinweise
	die Sammlungen und Datenbanken auflisten können, auf die sie zugreifen dürfen, ohne die jeweiligen `listDatabase` Rollen `listCollections` und zu benötigen • Benutzer können auch ihre eigenen Cursor beenden, ohne die `KillCursor` Rolle zu benötigen • Der Vergleich numerischer Typen von Unterdokumenten entspricht nun dem Vergleich numerischer Typen von Dokumenten der ersten Ebene. Das Verhalten in Amazon DocumentDB 4.0 ist jetzt mit MongoDB kompatibel. Fehlerbehebungen und andere Änderungen Amazon DocumentDB 4.0 (Engine-Patch-Version 2.0.722) • `$setOnInsert` erlaubt keine Aktualisierungen mehr, wenn der Positionsoperator verwendet wird. $ Das Verhalten in Amazon DocumentDB 4.0 ist jetzt mit MongoDB kompatibel. • Problem mit `$createCollection` und Set behoben `autoIndex Id` • Projektion für verschachtelte Dokumente • Die Standardeinstellung für den Arbeitsspeicher wurde geändert, sodass sie mit der Größe des Instanzspeichers skaliert wird • Verbesserungen bei der Speicherbereinigung • Suche mit leerem Schlüssel im Pfad, Verhaltensunterschied bei Mongo • `dateToString` Fehler im Zeitzonenverhalten behoben • Problem behoben `$push` (Aggregation), um die Sortierreihenfolge zu berücksichtigen • Fehler beim `$currentOp` Aggregieren behoben • Problem mit einem `readPreference` sekundären behoben • Es wurde ein Problem mit der Überprüfung `$createIndex` der gleichen Datenbank behoben, in der der Befehl ausgegeben wurde

Date	Versionshinweise
	• Das inkonsistente Verhalten bei der Suche schlägt `minKey` fehl `maxKey` • Problem behoben, bei dem der `$size` Operator nicht mit einem zusammengesetzten Array funktionierte • Problem mit der Negation von `$in` mit Regex behoben • Problem mit der `$distinct` Befehlsausführung für eine Ansicht behoben • Problem mit Aggregationen und Suchbefehlen, bei denen fehlende Felder unterschiedlich sortiert wurden, behoben • Problem behoben`$eq`, bei dem reguläre Ausdrücke den Typ nicht überprüften • `$currentDate` Fehler im Verhalten der Ordinalposition von Zeitstempeln behoben • Die Millisekunden-Granularität für wurde behoben `$currentDate`

Amazon DocumentDB 3.6

Versionshinweise für die Amazon DocumentDB-Engine Version 3.6 (MongoDB 3.6-Kompatibilität).

Date	Versionshinweise
31. August 2026	Neues Feature Amazon DocumentDB unterstützt jetzt direkte Hauptversions-Upgr ades (MVU) direkt von Engine-Version 3.6 auf Version 8.0. Dadurch sind keine Zwischenversions-Upgrades erforderlich und Sie können Ihre 3.6-Cluster unter Beibehaltung vorhandener Daten, Konfigura tionen und Cluster-Einstellungen aktualisieren. Dieses Upgrade ist in allen AWS Regionen verfügbar, in denen diese Versionen derzeit unterstützt werden. Weitere Informationen finden Sie unter Direktes Upgrade der Hauptversion von Amazon DocumentDB.
18. August 2026	Sicherheits- und Stabilitätsverbesserungen

Date	Versionshinweise
	Amazon DocumentDB 3.6 (Engine-Version: 1.0.208702) [in Bearbeitung] Dieser Engine-Patch bietet die folgenden Verbesserungen: • Verbesserte Anforderungsverarbeitung Amazon DocumentDB validiert eingehende Client-Anfragen jetzt gründlicher entlang des Verbindungs- und Nachrichtenanalysepfads. Dies verbessert die Widerstandsfähigkeit und Stabilität bei einer breiteren Palette von Eingabemustern. • Härtung des Authentifizierungs-Handshakes Amazon DocumentDB validiert den Authentifizierungsaustausch jetzt gründlicher. Dies verbessert die Robustheit beim Umgang mit unerwarteten oder falsch formatierten Eingaben. Wenn Sie ihn nicht früher anwenden, wendet Amazon DocumentDB diesen Patch automatisch während Ihres nächsten Wartungsfensters 30 Tage nach der Veröffentlichung an.
27. April 2026	Neue Features Amazon DocumentDB 3.6 (Maschinenversion: 1.0.208692) Dieses Update bietet einen optionalen Engine-Patch mit den folgenden Verbesserungen: 1. Unterstützung für `startAtOperationTime` Inchange-Streams hinzugefügt.

Date	Versionshinweise
12. Februar 2026	Fehlerbehebungen und andere Änderungen Amazon DocumentDB 3.6 (Engine-Version: 1.0.208675) 1. Die `DatabaseConnectionsActiveMax` Metrik zeichnet jetzt nach Instances auf. `Connections (active)` Weitere Informationen finden Sie unter Instanzkontingente. 2. Die Leistung der SASL-Authentifizierung wurde verbessert. 3. Fehlerbehebung für den Befehl $indexStats.
20. August 2024	Neues Feature Für Amazon DocumentDB 3.6 (Mindestversion des Engine-Patches 1.0.208662) ist für TLS-CA-Zertifikatsaktualisierungen kein Systemneustart mehr erforderlich. Weitere Informationen finden Sie unter Aktualisierung Ihrer Amazon DocumentDB-TLS-Zertifikate.
22. Juli 2024	Neue Features Amazon DocumentDB 3.6 (Engine-Patch-Version 1.0.208662) Indexbeschränkungen für db.r5.*- und db.r6.*-Instances in Amazon DocumentDB MVU wurden entfernt. Weitere Informationen finden Sie unter Voraussetzungen. Fehlerbehebungen und andere Änderungen Amazon DocumentDB 3.6 (Engine-Patch-Version 1.0.208662) Amazon DocumentDB erkennt jetzt -Nan als gültiges JSON-Token.
20. Dezember 2023	Weitere Änderungen Die Unterstützung für ein direktes Upgrade der Hauptversion in Amazon DocumentDB 3.6 wurde aktiviert. Weitere Informationen finden Sie unter Direktes Upgrade der Hauptversion von Amazon DocumentDB.

Date	Versionshinweise
17. November 2023	Fehlerbehebungen und andere Änderungen Amazon DocumentDB 3.6 (Engine-Patch-Version 1.0.208570) Benutzerdefinierte lokale Variablennamen unterstützen jetzt „_“ (Unterstrich) für Projektionsoperatoren wie und. `$let` `$filter`
20. September 2023	Neue Features Unterstützung für direkte Hauptversions-Upgrades in Amazon DocumentDB 3.6 wurde hinzugefügt. Weitere Informationen finden Sie unter Direktes Upgrade der Hauptversion von Amazon DocumentDB.
2. Februar 2023	Fehlerbehebungen und andere Änderungen Amazon DocumentDB 3.6 (Engine-Patch-Version 1.0.208432) • Es wurde ein Problem mit der DML-Überwachung behoben, wenn sie während einer laufenden Arbeitslast aktiviert wurde. • Problem mit der DML-Überwachung behoben, bei dem Aggregatb efehle mit Hinweis ein Zeichenfolgenwert übergeben wurden. • Es wurde ein Problem behoben, bei dem der `listCollections` Befehl nicht funktionierte, wenn Benutzer mit der Rolle readwrite anydatabase sowohl die Optionen authorizedCollections als auch NameOnly auf true gesetzt hatten. • Es gab ein Problem beim korrekten Analysieren einer numerischen Zeichenfolge in einem Feldnamen. Dieser Fehler wurde behoben. • Brechen Sie lang laufende Cursor ab, wenn sie sich auf die Speicherbereinigung auswirken.

Date	Versionshinweise
09. August 2022	Neue Features Amazon DocumentDB 3.6 (Engine-Patch-Version 1.0.208152) • Unterstützung für den Datentyp Decimal128 hinzugefügt. Der Decimal128 ist ein BSON-Datentyp, der in allen Regionen unterstützt wird, in denen DocumentDB verfügbar ist. Weitere Informationen finden Sie unter Data Types. • Unterstützung für die Prüfung von DML-Abfragen mit Amazon Logs wurde hinzugefügt. CloudWatch Jetzt kann Amazon DocumentDB DML-Ereignisse (Data Manipulation Language) und DDL-Ereignisse (Data Definition Language) in Amazon Logs aufzeichnen. CloudWatch Weitere Informationen finden Sie in diesem Blog-Beitrag. Fehlerbehebungen und andere Änderungen Amazon DocumentDB 3.6 (Engine-Patch-Version 1.0.208152) • Sie können jetzt Ihr eigenes Passwort mit einem eigenen Passwort mit Privilegien ändern. `changeOwnPassword`
7. April 2022	Neue Features Amazon DocumentDB 3.6 (Engine-Patch-Versionen 1.0.207836 und 1.0.208015) Amazon DocumentDB Performance Insights befindet sich jetzt in der Vorschauversion. Sie können jetzt den Leistungsverlauf von sieben Tagen ohne zusätzliche Kosten in einem fortlaufenden Fenster speichern. Weitere Informationen finden Sie unter Überwachung mit Performance Insights.

Date	Versionshinweise

24. Januar 2022	Neue Features
	
	Amazon DocumentDB 3.6 (Engine-Patch-Version 1.0.207684)
	
	• Amazon DocumentDB bietet jetzt eine kostenlose Testversion an. Einzelheiten finden Sie auf der kostenlosen Testseite von Amazon DocumentDB.
	• Sie können jetzt erweiterte Funktionen mit Geospatial-Abfragen verwenden, einschließlich der folgenden APIs:
	  • `$geoWithin`
	  • `$geoIntersects`
	• Unterstützung für die folgenden MongoDB-Operatoren wurde hinzugefügt:
	  • `$mergeObjects`
	  • `$reduce`
	
	Weitere Informationen finden Sie unter Abfragen von Geodaten mit Amazon DocumentDB.

21. Januar 2022	Neue Features
	
	Amazon DocumentDB 3.6 (Engine-Patch-Version 1.0.207781)
	
	• Unterstützung für die folgenden MongoDB-APIs wurde hinzugefügt:
	  • `$reduce`
	  • `$mergeObjects`
	  • `$geoWithin`
	  • `$geoIntersects`

Date	Versionshinweise

25. Oktober 2021	Neue Features
	
	Amazon DocumentDB 3.6 (Engine-Patch-Version 1.0.207780)
	
	• Unterstützung für die folgenden MongoDB-APIs wurde hinzugefügt
	  • `$literal`
	  • `$map`
	  • `$$ROOT`
	• Unterstützung für GeoSpatial Abfragefunktionen. Weitere Informationen finden Sie in diesem Blogbeitrag
	• Unterstützung für die Zugriffskontrolle mit benutzerdefinierten Rollen. Weitere Informationen finden Sie in diesem Blogbeitrag
	• Amazon DocumentDB JDBC-Treiber, um die Konnektivität von BI-Tools wie Tableau und Abfragetools wie SQL Workbench zu ermöglichen
	
	Fehlerbehebungen und andere Änderungen
	
	Amazon DocumentDB 3.6 (Engine-Patch-Version 1.0.207780)
	
	• Fehlerkorrektur für die korrekte Sortierung`$natural`, wenn ein explizites `.sort()` Zeichen zusammen mit vorhanden ist `$natural`
	• Bugfix für Change Stream to work with `$redact`
	• Bugfix für `$ifNull` die Arbeit mit leerem Array
	• Bugfix für übermäßigen consumption/server Ressourcenabsturz, wenn ein aktuell angemeldeter Benutzer gelöscht wurde oder diesem Benutzer die Berechtigung für eine laufende Aktivität entzogen wurde
	• Fehlerkorrektur und Rechteüberprüfung `listDatabase listCollection`
	• Fehler behoben: Deduplizierungslogik für Elemente mit mehreren Schlüsseln

Date	Versionshinweise
24. Juni 2021	Neue Features Amazon DocumentDB 3.6 (Engine-Patch-Version 1.0.207117) • r5.8xlarge- und r5.16xlarge Instances werden jetzt unterstützt. Weitere Informationen finden Sie im Blogbeitrag Amazon DocumentDB unterstützt jetzt die Instances r5.8xlarge und r5.16xlarge. • Globale Cluster werden jetzt unterstützt, um Disaster Recovery nach Region-wide Ausfällen zu ermöglichen und globale Lesevorgänge mit niedriger Latenz zu ermöglichen, indem Lesevorgänge aus dem nächstgelegenen Amazon DocumentDB-Cluster zugelassen werden. Beachten Sie, dass globale Cluster derzeit in den Regionen Südamerika (São Paulo), Europa (Mailand), China (Peking) und China (Ningxia) nicht unterstützt werden.

Date	Versionshinweise
4. Mai 2021	Neue Features Amazon DocumentDB 3.6 (Engine-Patch-Version 1.0.207117) • `renameCollection` • `$zip` • `$indexOfArray` • `$reverseArray` • `$natural` • `$hint`Unterstützung für Updates • Index-Scan für `distinct` Fehlerbehebungen und andere Änderungen Amazon DocumentDB 3.6 (Engine-Patch-Version 1.0.207117) • Reduzierter Speicherverbrauch für Abfragen `$in` • Ein Speicherverlust in Multikey-Indizes wurde behoben • Der Erklärungsplan und die Profiler-Ausgabe für wurden korrigiert `$out` • Ein Timeout für Operationen aus dem internen Überwachu ngssystem wurde hinzugefügt, um die Zuverlässigkeit zu verbessern • Es wurde ein Fehler behoben, der sich auf die Abfrageprädikate auswirkte, die an Multikey-Indizes übergeben wurden

Date	Versionshinweise
15. Januar 2021	Neue Features Amazon DocumentDB 3.6 (Engine-Patch-Version 1.0.206295) • Fähigkeit, einen Index mit der Aggregationsphase zu verwenden `$lookup` • `find()`Abfragen mit Projektionen können direkt von einem Index aus bedient werden (abgedeckte Abfrage) • Fähigkeit zur Verwendung `hint()` mit `findAndModify` • Leistungsoptimierungen für `$addToSet` den Bediener • Verbesserungen zur Reduzierung der Gesamtindexgrößen • Neue Aggregationsoperatoren: `$ifNull``$replaceRoot` ,`$setIsSubset` , `$setInstersection` `$setUnion` , und `$setEquals` • Benutzer können auch ihre eigenen Cursor beenden, ohne die Rolle zu benötigen `KillCursor`
9. November 2020	Neue Features Amazon DocumentDB 3.6 (Engine-Patch-Version 1.0.206295) • Keine Fehlerbehebungen und andere Änderungen Amazon DocumentDB 3.6 (Engine-Patch-Version 1.0.206295) • Keine

Date	Versionshinweise
30. Oktober 2020	Neue Features Amazon DocumentDB 3.6 (Engine-Patch-Version 1.0.206295) • Es wurde die Möglichkeit hinzugefügt, einen Change-Stream-Cursor auf Cluster-Ebene (`client.watch()` oder in der Datenbank zu öffnen `mongo.watch()) (db.watch())` • Möglichkeit, die Aufbewahrungsfrist für Change-Streams auf 7 Tage (zuvor 24 Stunden) zu erhöhen Fehlerbehebungen und andere Änderungen Amazon DocumentDB 3.6 (Engine-Patch-Version 1.0.206295) • Verschiedene allgemeine Leistungsverbesserungen • Eine gezielte Sicherheitsverbesserung • Ein Problem mit dem Überspringen der Sortierung im zweiten Feld eines zusammengesetzten Indexes wurde behoben • Aktiviert den regulären Index für Gleichheit in einem einzelnen Feld eines Indexes mit mehreren Schlüsseln (nicht zusammengesetzt) • Die Startbedingung für die Authentifizierung wurde behoben • Problem behoben, das zu einem seltenen Absturz der Garbage-C ollection führte • Verbesserung der RBAC-Sicherheit • Metrik hinzugefügt `databaseConnectionsMax` • Leistungsverbesserungen für bestimmte Workloads auf Instances `r5.24xlarge`

Date	Versionshinweise
22. September 2020	Neue Features Amazon DocumentDB 3.6 (Engine-Patch-Version 1.0.206295) • `$out`Aggregationsphase • Die maximale Anzahl von Verbindungen und Cursor pro Instanz wurde um das 10-fache erhöht Fehlerbehebungen und andere Änderungen Amazon DocumentDB 3.6 (Engine-Patch-Version 1.0.206295) • Keine
10. Juli 2020	Neue Features Amazon DocumentDB 3.6 (Engine-Patch-Version 1.0.206295) • Regionsübergreifende Snapshot-Kopie Fehlerbehebungen und andere Änderungen Amazon DocumentDB 3.6 (Engine-Patch-Version 1.0.206295) • Keine

Date	Versionshinweise
30. Juni 2020	Neue Features Amazon DocumentDB 3.6 (Engine-Patch-Version 1.0.206295) • Mittlere T3-Instanzen Fehlerbehebungen und andere Änderungen Amazon DocumentDB 3.6 (Engine-Patch-Version 1.0.206295) • Rückgewinnung von ungenutztem Speicher für t3-Instances • Verbesserungen bei der Authentifizierung • Verbesserte Leistung bei der SASL-Authentifizierung • `currentOp` Problem behoben, bei dem die maximal möglichen Operationen überschritten wurden • `killOps`Problem beim Massenupdate und Löschen behoben • `$sample`Leistungsverbesserungen mit `$match` • Die Unterstützung für den zweiten Fall $$ in der Redigierungsphase wurde behoben • Verschiedene wiederkehrende Absturzursachen wurden behoben • Verbesserungen am TTL-Sweeping zur Reduzierung von iOS und Latenz • Optimierte Speichernutzung für `$unwind` • Die Startbedingung für Sammlungsstatistiken mit Drop-Index wurde behoben • Die Rennbedingung beim gleichzeitigen Erstellen des Indexes wurde behoben • Seltener Absturz im `hash_search` Index behoben

Allgemeine Versionshinweise zu Amazon DocumentDB

Allgemeine Serviceupdates, einschließlich regionaler Verfügbarkeit, Tool-Integrationen und versionsübergreifender Funktionen.

Date	Versionshinweise
15. September 2026	Neues Feature Der serverlose Amazon DocumentDB-Instance-Typ ist jetzt in den Amazon DocumentDB-Engine-Versionen 5.0 und 8.0 in den folgenden Regionen verfügbar: • Asien-Pazifik (Osaka) • Asien-Pazifik (Jakarta) • Asien-Pazifik (Malaysia) • Asien-Pazifik (Melbourne) • Asien-Pazifik (Thailand) • Kanada West (Calgary) • Europa (Zürich) • Europa (Stockholm) • Israel (Tel Aviv) • Mexiko (Zentral) Weitere Informationen erhalten Sie unter Unterstützte Instance-Klassen nach Region und Serverlose Verwendung von Amazon DocumentDB.
29. April 2026	Neues Feature Amazon DocumentDB ist jetzt in der Region Kanada West (Calgary) verfügbar. Weitere Informationen finden Sie in diesem Blog-Beitrag.
17. April 2026	Neues Feature Eine schnellere Skalierung auf DocumentDB serverless ist jetzt verfügbar .

Date	Versionshinweise
11. Februar 2026	Neues Feature Amazon DocumentDB ist jetzt in der Region Asien-Pazifik (Melbourne) verfügbar. Weitere Informationen finden Sie in diesem Blog-Beitrag. Amazon DocumentDB ist jetzt in der Region Europa (Zürich) verfügbar. Weitere Informationen finden Sie in diesem Blog-Beitrag.
08. Januar 2026	Neues Feature Amazon DocumentDB ist jetzt in der Region Asien-Pazifik (Jakarta) verfügbar. Weitere Informationen finden Sie in diesem Blog-Beitrag.
13. Oktober 2025	Neues Feature Amazon DocumentDB unterstützt jetzt R6G- und T4G-Instances in der Region Asien-Pazifik (Hyderabad). Weitere Informationen finden Sie unter Unterstützte Instance-Klassen nach Region.
7. Oktober 2025	Neues Feature Amazon DocumentDB ist jetzt in den folgenden Regionen verfügbar: • Asien-Pazifik (Malaysia) • Asien-Pazifik (Osaka) • Asien-Pazifik (Thailand) • Mexiko (Zentral) Weitere Informationen finden Sie in diesem Blog-Beitrag.

Date	Versionshinweise
26. September 2025	Neue Features Amazon DocumentDB unterstützt jetzt das regions- und kontoüber greifende Kopieren und Teilen von Snapshots. Weitere Informationen erhalten Sie unter Schnappschüsse kopieren AWS-Regionen und Amazon DocumentDB-Cluster-Snapshots teilen.
18. Juni 2025	Neues Feature Amazon DocumentDB ist jetzt in der Region Israel (Tel Aviv) verfügbar.
8. Mai 2025	Neues Feature Amazon DocumentDB ist jetzt in der Region Europa (Stockholm) verfügbar. Weitere Informationen finden Sie in diesem Blog-Beitrag.
2. April 2025	Fehlerbehebungen und andere Änderungen Ein Fehler bei der Erstellung von Vektorindizes mit Hintergrundindizes wurde behoben.
6. Februar 2025	Neues Feature Amazon DocumentDB ist jetzt integriert in. AWS Toolkit for Visual Studio Code Weitere Informationen finden Sie in diesem Blogbeitrag „Was ist neu“ und im Benutzerhandbuch unter Arbeiten mit Amazon DocumentDB im AWS Toolkit for Visual Studio Code Toolkit.
22. Oktober 2024	Neues Feature Amazon DocumentDB unterstützt jetzt Messwerte für den Durchsatz von Speichernetzwerken `StorageNetworkReceiveThroughput` und`StorageNetworkTransmitThroughput` . `StorageNetworkThroughput` Weitere Informationen finden Sie unter Bewertung der Nutzung der Amazon DocumentDB-Instance anhand von Metriken CloudWatch.

Date	Versionshinweise
18. September 2024	Neues Feature Amazon DocumentDB ist jetzt in der Region Afrika (Kapstadt) verfügbar. Weitere Informationen finden Sie in diesem Blog-Beitrag. Amazon DocumentDB ist jetzt in der Region Europa (Spanien) verfügbar. Weitere Informationen finden Sie in diesem Blog-Beitrag.
25. Juni 2024	Neues Feature Die Authentifizierung mit AWS IAM-ARNs ist in auf Amazon DocumentDB-Instances basierenden 5.0-Clustern in allen unterstützten Regionen verfügbar. Weitere Informationen finden Sie unter Authentifizierung mithilfe der IAM-Identität.
3. April 2024	Neues Feature Amazon DocumentDB ist jetzt in der Region Naher Osten (VAE) verfügbar. Weitere Informationen finden Sie in diesem Blog-Beitrag. Fehlerbehebungen und andere Änderungen • Ein Fehler beim Aufrufen `db.coll.stats()` auf der Mongo-Shell-Version 1.7 und höher wurde behoben. • Ein Speicherverlust bei Change-Stream-Abfragen, die `$regex` als Teil derselben Aggregationspipeline enthalten, wurde behoben.
30. Januar 2024	Neue Features Globale Amazon DocumentDB-Cluster Globale Cluster sind jetzt in beiden AWS GovCloud (US) Regionen verfügbar: AWS GovCloud (US-East) und AWS GovCloud (US-West).

Date	Versionshinweise
13. Dezember 2023	Neue Features Unterstützung für 1-Click-EC2-Konnektivität hinzugefügt. Weitere Informationen finden Sie unter Stellen Sie mithilfe von Amazon EC2 eine Verbindung her.
17. November 2023	Neue Features Amazon DocumentDB ist jetzt in der Region AWS GovCloud () US-East verfügbar. Weitere Informationen finden Sie in diesem Blog-Beitrag.
25. September 2023	Neue Features Amazon DocumentDB ist jetzt in der Region Asien-Pazifik (Hongkong) verfügbar. Weitere Informationen finden Sie in diesem Blog-Beitrag.
11. September 2023	Neue Features Amazon DocumentDB ist jetzt in der Region Asien-Pazifik (Hyderabad) verfügbar. Weitere Informationen finden Sie in diesem Blog-Beitrag.
29. April 2022	Neue Features Amazon DocumentDB ist jetzt in der Region China (Peking) verfügbar. Weitere Informationen finden Sie in diesem Blog-Beitrag.
16. März 2022	Neue Features Amazon DocumentDB ist jetzt in der Region Europa (Mailand) verfügbar. Weitere Informationen finden Sie in diesem Blog-Beitrag.
8. Februar 2022	Neue Features Amazon DocumentDB R6G- und T4G-Instances sind jetzt im asiatisch-pazifischen Raum, Südamerika und Europa verfügbar. Weitere Informationen finden Sie in diesem Blog-Beitrag.

Date	Versionshinweise
4. Mai 2021	Neue Features In diesem Blogbeitrag finden Sie alle neuen Funktionen. https://aws.amazon.com/about-aws/whats-new/2021/05/amazon-documentdb-improves-mongodb-compatibility-and-indexing-improvements/
9. November 2020	Neue Features Alle neuen Funktionen finden Sie in diesem Blogbeitrag.
30. Oktober 2020	Neue Features Alle neuen Funktionen finden Sie in diesem Blogbeitrag.
22. September 2020	Neue Features Alle neuen Funktionen finden Sie in diesem Blogbeitrag.
10. Juli 2020	Neue Features Alle neuen Funktionen finden Sie in diesem Blogbeitrag.
30. Juni 2020	Neue Features Alle neuen Funktionen finden Sie in diesem Blogbeitrag.

Versionshinweise zu Amazon DocumentDB Elastic Clusters

Versionshinweise für Amazon DocumentDB Elastic Clusters.

Date	Versionshinweise
28. Januar 2025	Neues Feature Für Konnektivität mit AWS CloudShell nur einem Klick ist Amazon DocumentDB jetzt in vier instanzbasierte und elastische Cluster integrier t.

Date	Versionshinweise
	Weitere Informationen finden Sie in einigen oder allen der folgenden Links: • Erste Schritte mit Amazon DocumentDB • Erste Schritte mit Amazon DocumentDB Elastic Clusters • Was ist ein neuer Blogbeitrag • Blogbeitrag mit technischen Anleitungen
12. November 2024	Neue Features Elastische Amazon DocumentDB-Cluster unterstützen jetzt Hintergrundindizes.
1. November 2024	Neues Feature Amazon DocumentDB unterstützt jetzt Wartungsaktionen für elastische Cluster. Weitere Informationen finden Sie unter Wartung der elastischen Amazon DocumentDB-Cluster.
8. August 2024	Neues Feature Elastische Amazon DocumentDB-Cluster sind jetzt in den Regionen Asien-Pazifik (Hongkong), Kanada (Zentral) und Europa (Paris) verfügbar. Weitere Informationen finden Sie unter Regionale Verfügbarkeit für elastische Cluster.
8. Juli 2024	Neues Feature Elastische Amazon DocumentDB-Cluster sind jetzt in der Region Europa (Mailand) verfügbar. Weitere Informationen finden Sie unter Regionale Verfügbarkeit für elastische Cluster.

Date	Versionshinweise
22. Februar 2024	Neue Features Elastische Amazon DocumentDB-Cluster Elastische Amazon DocumentDB-Cluster unterstützen jetzt die folgenden Funktionen: • Lesbare Replikate von sekundären Shard-Instances — weitere Informationen finden Sie in Schritt 5b von. Schritt 1: Erstellen Sie einen elastischen Cluster • Start/stop Cluster — weitere Informationen finden Sie unter. Stoppen und Starten eines elastischen Amazon DocumentDB-Clusters • Konfigurierbare Shard-Instances — weitere Informationen finden Sie in Schritt 5b vonSchritt 1: Erstellen Sie einen elastischen Cluster. • Automatische Backups für Snapshots — weitere Informationen finden Sie unter. Verwaltung eines automatischen Backups eines Elastic Cluster-Snapshots • Snapshot kopieren — weitere Informationen finden Sie unterEinen Elastic Cluster-Snapshot kopieren.
30. Januar 2024	Neue Features Elastische Amazon DocumentDB-Cluster Elastische Amazon DocumentDB-Cluster sind jetzt in den folgenden Regionen verfügbar: • Asien-Pazifik (Mumbai) • Asien-Pazifik (Seoul) • Südamerika (São Paulo) • Europa (London) Weitere Informationen finden Sie unter Verfügbarkeit von Regionen und Versionen des Elastic Clusters.

Date	Versionshinweise

3. August 2023	Neue Features
	
	Amazon DocumentDB Elastic-Cluster
	
	• Amazon DocumentDB Elastic-Cluster unterstützen jetzt die folgenden Operationen:
	  • `top`
	  • `collStats`
	  • `hint`
	  • `dataSize`
	
	  Eine vollständige Liste der unterstützten Befehle und Operationen finden Sie unter. Unterstützte MongoDB-APIs, Operationen und Datentypen in Amazon DocumentDB
	• Time-to-Live-Indizes (TTL) werden jetzt unterstützt.
	• Indizes `hints` werden jetzt mit Indexausdrücken unterstützt.

22. März 2023	Neue Features Elastische Amazon DocumentDB-Cluster sind jetzt in den Regionen Asien-Pazifik (Singapur), Asien-Pazifik (Sydney) und Asien-Pazifik (Tokio) verfügbar. Weitere Informationen finden Sie unter Verfügbarkeit von Regionen und Versionen des Elastic Clusters.

Date	Versionshinweise
30. November 2022	Neue Features Elastische Amazon DocumentDB-Cluster Amazon DocumentDB Elastic Clusters ist eine neue Art von Amazon DocumentDB-Cluster, mit dem Benutzer die MongoDB-Sharding-APIs nutzen können, um ihren Cluster zu skalieren. Elastische Cluster verarbeiten praktisch eine beliebige Anzahl von Lese- und Schreibvorgängen mit Petabytes an Speicherkapazität, indem sie die Daten und Berechnungen auf mehrere zugrunde liegende Recheninstanzen und Volumes verteilen. Weitere Informationen finden Sie unter Elastic Clusters von Amazon DocumentDB verwenden.

Dokumentenverlauf für das Amazon DocumentDB Developer Guide

- API-Version: 2014-10-31
- Letzte Aktualisierung der Dokumentation: 27. Mai 2026

In der folgenden Tabelle wird die Dokumentation für diese Version des Amazon DocumentDB Developer Guide beschrieben.

Änderung	Beschreibung	Datum
Neues Amazon DocumentDB-Entwicklerhandbuch	Das Amazon DocumentDB-Entwicklerhandbuch ist jetzt unter einer neuen URL veröffentlicht. Die API-Referenz ist separat unter Amazon DocumentDB API Reference verfügbar.	27. Mai 2026
AWS verwaltetes Richtlinienupdate — Änderung der Richtlinie	Amazon DocumentDB aktualisiert die Richtlinien für vollen Zugriff für elastische Cluster.	11. Februar 2025
AWS verwaltete Aktualisierung der Richtlinie — Änderung der Richtlinie	Amazon DocumentDB aktualisiert die Richtlinien für vollen Zugriff für elastische Cluster.	21. Februar 2024
AWS verwaltete Richtlinienaktualisierung — Änderung der Richtlinie	Amazon DocumentDB aktualisiert die Richtlinien für Schreibschutz und Vollzugriff für elastische Cluster.	21. Juni 2023
AWS verwaltetes Richtlinien-Update — neue Richtlinie	Amazon DocumentDB führt eine neue schreibgeschützte	08. Juni 2023

	Richtlinie für elastische Cluster ein.	
AWS verwaltetes Richtlinien-Update — neue Richtlinie	Amazon DocumentDB führt eine neue Richtlinie für vollständigen Zugriff für elastische Cluster ein.	5. Juni 2023
MongoDB 5.0-Kompatibilität	Amazon DocumentDB ist jetzt mit Version 5.0 von MongoDB kompatibel.	1. März 2023
Aktualisierung der Richtlinie	Zur Unterstützung der Elastic-Cluster-Funktion von Amazon DocumentDB AmazonDocDB-ElasticServiceRolePolicy wird die AmazonDocDBConsole FullAccess Richtlinie aktualisiert und eingeführt.	30. November 2022
Elastische Cluster	Neue Elastic Cluster-Funktion hinzugefügt, die die Hash-basierte Partitionierung (Sharding) von Daten im verteilten Speichersystem von Amazon DocumentDB unterstützt.	30. November 2022
Globale Cluster	Dokumentation zur Verwendung von Global Clusters hinzugefügt.	2. Juni 2021
Abonnemente für Veranstaltungen	Dokumentation zum Veranstaltungsabonnement hinzugefügt.	26. März 2021

Aktualisierungen auf Version 3.6	Dokumentierte Verbesserungen gegenüber Version 3.6 in Bezug auf rollenbasierte Zugriffskontrollen, Aggregationsoperatoren und Leistung.	15. Januar 2021
MongoDB 4.0-Kompatibilität	Amazon DocumentDB ist jetzt mit Version 4.0 von MongoDB kompatibel.	9. November 2020
Anleitungen für den Einstieg	Neue Anleitungen für die ersten Schritte mit Amazon DocumentDB mithilfe von Amazon EC2 AWS Cloud9, Robo3T oder Studio3T.	15. August 2020
Zusätzliche Availability Zones werden unterstützt	Amazon DocumentDB hat Unterstützung für eine zusätzliche Availability Zone im asiatisch-pazifischen Raum (Seoul) hinzugefügt (ap-northeast-2).	14. Juli 2020
Unterstützung für das Kopieren von Schnappschüssen zwischen Regionen hinzugefügt.	Amazon DocumentDB hat Unterstützung für das Kopieren von Cluster-Snapshots hinzugefügt. AWS-Regionen Weitere Informationen finden Sie unter Regionsübergreifendes Kopieren von Snapshots.	10. Juli 2020

Unterstützung für die T3-Instance-Klasse hinzugefügt.	Unterstützung für T3-Instance-Typen in allen Regionen hinzugefügt, die Amazon DocumentDB unterstützen. Weitere Informationen finden Sie unter Unterstützte Instance-Klassen nach Region und Instance-Klassenspezifikationen.	30. Juni 2020
Unterstützung für hinzugefügt AWS GovCloud (US).	Amazon DocumentDB ist jetzt in der AWS GovCloud (US) Region (us-gov-west-1) verfügbar.	29. Juni 2020
16 neue Metriken hinzugefügt. CloudWatch	Amazon DocumentDB hat Unterstützung für 16 neue CloudWatch Amazon-Metriken hinzugefügt. Weitere Informationen finden Sie unter Überwachung von Amazon DocumentDB mit. CloudWatch	23. Juni 2020

Unterstützung für Nullzeichen und den $regex-Operator hinzugefügt.	Amazon DocumentDB hat Unterstützung für Nullzeichen in Zeichenketten und die Möglichkeit hinzugefügt, einen Index für $regex zu verwenden. Informationen zu den unterstützten MongoDB-APIs und Aggregationspipeline-Funktionen für Amazon DocumentDB finden Sie unter Funktionale Unterschiede zu MongoDB. https://docs.aws.amazon.com/documentdb/latest/devguide/functional-differences.html	22. Juni 2020
Unterstützung für verbesserte Indexierungsfunktionen mit mehreren Schlüsseln wurde hinzugefügt.	Amazon DocumentDB hat die Unterstützung für verbesserte Multi-Key-Indexierungsfunktionen hinzugefügt, die die Indizierung von Arrays mit mehr als 2.048 Byte umfassen, und die Möglichkeit, einen zusammengesetzten Mehrschlüsselindex mit mehreren Schlüsseln in demselben Array zu erstellen. Weitere Informationen finden Sie unter Funktionsunterschiede bei MongoDB.	23. April 2020

Unterstützung für den Löschschutz für einen Amazon DocumentDB-Stack hinzugefügt. CloudFormation	Amazon DocumentDB hat Unterstützung für die Aktivierung des Löschschutzes beim Erstellen eines Amazon CloudFormation DocumentDB-Stacks hinzugefügt.	20. April 2020
Unterstützung für rollenbasierte Zugriffskontrolle hinzugefügt.	Amazon DocumentDB hat Unterstützung für rollenbasierte Zugriffskontrolle mithilfe integrierter Rollen hinzugefügt.	26. März 2020
Unterstützung für eine zusätzliche Availability Zone in Kanada (Central) hinzugefügt (ca-central-1).	Amazon DocumentDB ist jetzt in der Region Kanada (Central) (ca-central-1) mit Instances der Klasse R5 und 3 Availability Zones verfügbar.	26. März 2020
Unterstützung für zwei zusätzliche MongoDB-APIs hinzugefügt.	Amazon DocumentDB hat Unterstützung für `$dateFromString` und `executionStats` MongoDB-APIs hinzugefügt.	23. März 2020
Unterstützung für fünf weitere MongoDB-APIs hinzugefügt.	Amazon DocumentDB hat Unterstützung für`$objectToArray` , `$arrayToObject` `$slice`$mod, und `$range` MongoDB-APIs hinzugefügt.	6. Februar 2020
Unterstützung für Kanada (Central) hinzugefügt.	Amazon DocumentDB ist jetzt in der Region Kanada (Central) (ca-central-1) mit Instances der Klasse R5 verfügbar.	11. Dezember 2019

ChangeStreamLogSizeUnterstützung für hinzugefügt.	Amazon DocumentDB hat Unterstützung `ChangeStreamLogSize` für Cloudwatch-Metriken hinzugefügt.	22. November 2019
Unterstützung für die Region Europa (Paris) hinzugefügt	Amazon DocumentDB ist jetzt in der Region Europa (Paris) (eu-west-3) mit Instances der Klasse R5 verfügbar.	30. Oktober 2019
Unterstützung für die Region Asien-Pazifik (Mumbai) hinzugefügt	Amazon DocumentDB ist jetzt in der Region Asien-Pazifik (Mumbai) (ap-south-1) mit Instances der Klasse R5 verfügbar.	17. Oktober 2019
Unterstützung für drei zusätzliche MongoDB-APIs hinzugefügt	Amazon DocumentDB hat Unterstützung für die APIs`$addFields` ,`$concatArrays` , und `$lookup` MongoDB hinzugefügt.	16. Oktober 2019
Unterstützung für die Region Asien-Pazifik (Singapur) hinzugefügt	Amazon DocumentDB ist jetzt in der Region Asien-Pazifik (Singapur) (ap-southeast-1) mit Instances der Klasse R5 verfügbar.	14. Oktober 2019
Neues Dokument zur Aktualisierung von TLS-Zertifikaten hinzugefügt	Anweisungen zum Aktualisieren von Zertifizierungsstellenzertifikaten zur Verwendung des neuen Zertifizierungsstellenzertifikats zum Erstellen von TLS-Verbindungen wurden hinzugefügt.	2. Oktober 2019

API-Unterstützung für Zertifikate hinzugefügt	Amazon DocumentDB ist ein neuer Zertifikatsdatentyp für Instances. Weitere Informationen finden Sie unter DBInstance.	1. Oktober 2019
Unterstützung für die Erstellung von Abfrageprofilen	Amazon DocumentDB hat die Möglichkeit hinzugefügt, unterstützte Operationen auf den Instances und Datenbanken Ihres Clusters zu profilieren.	19. August 2019
Dritter AZ im asiatisch-pazifischen Raum (Tokio) hinzugefügt	Amazon DocumentDB hat eine dritte Availability Zone (AZ) für Ihre Recheninstanzen im asiatisch-pazifischen Raum (Tokio) hinzugefügt.	9. August 2019
Unterstützung für zusätzliche Mongo-APIs	Unterstützung für zusätzliche Aggregationspipeline-Funktionen hinzugefügt, zu denen die Aggregationsoperatoren `$in` `$isoWeek` `$isoWeekYear` ,`$isoDayOfWeek` ,, und die `$dateToString` `$addToSet` Aggregationsphase gehören. Amazon DocumentDB hat außerdem Unterstützung für den `top()` Befehl zur Diagnose auf Sammlungsebene und die Möglichkeit hinzugefügt, den `expireAfterSeconds` Parameter für TTL-Indizes mithilfe des Befehls zu ändern. `collMod()`	31. Juli 2019

Unterstützung für Europa (London) hinzugefügt	Amazon DocumentDB ist jetzt in Europa (London) (eu-west-2) mit Instances der Klasse R5 verfügbar.	18. Juli 2019
Codebeispiele hinzugefügt	Codebeispiele in R und Ruby für die programmgesteuerte Verbindung zu Amazon DocumentDB hinzugefügt.	17. Juli 2019
Best Practice hinzugefügt	Eine bewährte Methode wurde hinzugefügt, die Ihnen bei der Verwaltung Ihrer Amazon DocumentDB-Kosten hilft.	17. Juli 2019
Unterstützung für das Stoppen und Starten eines Clusters	Amazon DocumentDB hat Unterstützung für das Stoppen und Starten von Clustern hinzugefügt, um die Kosten für Entwicklungs- und Testumgebungen zu verwalten.	1. Juli 2019

Unterstützung für den Schutz vor dem Löschen von Clustern	Um Ihre Cluster vor versehentlichem Löschen zu schützen, hat Amazon DocumentDB einen Löschschutz hinzugefügt. Weitere Informationen finden Sie in den folgenden Themen: Erstellen eines Amazon DocumentDB-Clusters, Ändern eines Amazon DocumentDB-Clusters, Löschen eines Amazon DocumentDB-Clusters und `DeletionProtection` im API-Thema dbCluster. https://docs.aws.amazon.com/documentdb/latest/APIReference/API_DBCluster.html	1. Juli 2019
Aktualisierung der funktionalen Unterschiede	Implizite Transaktionen wurden den Funktionsunterschieden hinzugefügt.	26. Juni 2019
Ergänzung zu funktionellen Unterschieden	Hinweis zur Speicherung und Indexkomprimierung in Amazon DocumentDB hinzugefügt.	13. Juni 2019
Zusätzliche Region wird unterstützt	Amazon DocumentDB ist jetzt im asiatisch-pazifischen Raum (Sydney) (ap-southeast-2) mit Instances der Klasse R5 verfügbar.	5. Juni 2019

Die R5-Instance-Klasse wird in weiteren Regionen unterstützt	Unterstützung der R5-Instance-Klasse für 4 weitere Regionen hinzugefügt: USA Ost (Ohio), USA Ost (Nord-Virginia), USA West (Oregon) und EU (Irland). Mit dieser Änderung werden R5-Instances in allen Regionen unterstützt, die Amazon DocumentDB unterstützen.	17. Mai 2019
Zusätzliche Regionen werden unterstützt	Unterstützung für zwei weitere Regionen hinzugefügt, Asien-Pazifik (Tokio) (ap-northeast-1) und Asien-Pazifik (Seoul) (ap-northeast-2) mit R5-Instance-Klassen. Weitere Informationen finden Sie unter Unterstützte Instance-Klassen nach Region und Instance-Klassen-Spezifikationen.	8. Mai 2019
Weitere Beispiele für Verbindungscodes hinzugefügt	Codebeispiele in Java und C# für die Verbindung zu Amazon DocumentDB hinzugefügt.	24. April 2019

Zusätzliche Mongo-API-Unterstützung	Es wurde die Unterstützung von sieben Aggregations-Zeichenfolgenoperatoren (`$indexOfBytes`, `$indexOfCP`, `$strLenBytes`, `$strLenCP`, `$toLower`, `$toUpper`, `$split`), neun Datumszeitoperatoren (`$dayOfYear`, `$dayOfMonth`, `$dayOfWeek`, `$year`, `$month`, `$hour`, `$minute`, `$second`, `$millisecond`) und die Aggregations-Pipeline-Stufe `$sample` hinzugefügt.	4. April 2019
Beispiele für Verbindungscodes hinzugefügt	Codebeispiele in Python Node.js, PHP und Go für die Verbindung mit Amazon DocumentDB hinzugefügt.	21. März 2019
Unterstützung für Frankfurt Region und R5-Instances	Unterstützung für die Region Europa (Frankfurt) (eu-central-1) mit R5-Instance-Klassen hinzugefügt. Weitere Informationen finden Sie unter Unterstützte Instance-Klassen nach Region und Instance-Klassen-Spezifikationen.	13. März 2019

Unterstützung für Betreiber von Aggregations-Pipelines	Es wurde die Unterstützung für neue Aggregations-Zeichenfolgenoperatoren (`$concat`, `$substr`, `$substrBytes`, `$substrCP`, `$strcasecmp`), einen Array-Aggregationsoperator (`$size`), einen Aggregationsgruppen-Akkumulatoroperator (`$push`) und Aggregationsstufen (`$redact` und `$indexStats`) hinzugefügt. Wir haben außerdem die Unterstützung für Positional-Array-Operatoren (`$[]` und `$[<identifier>]`) und `hint()` hinzugefügt.	28. Februar 2019
Motor-Upgrades	Es wurde eine Dokumentation zur Ermittlung ausstehender Cluster-Änderungen und Aktualisierung der Engine Version Ihres Clusters hinzugefügt.	15. Februar 2019
Ereignisse werden überwacht	Unterstützung für die Überwachung von Datenbankereignissen mit Amazon CloudWatch Logs hinzugefügt.	12. Februar 2019
Schnellstart	Es wurde ein Schnellstart-Thema hinzugefügt, das Ihnen den Einstieg in die Verwendung CloudFormation von Amazon DocumentDB erleichtert.	11. Januar 2019

Öffentliche Veröffentlichung	Dies ist die erste öffentliche Version von Amazon DocumentDB (mit MongoDB-Kompatibilität). Diese Version enthält den Entwicklerleitfaden und die integrierte API-Referenz für das Ressourcenmanagement.	9. Januar 2019

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.